# Reachability and Connectivity Queries in Constraint Databases*

**Michael Benedikt**† Bell Laboratories
**Martin Grohe**‡ Univ. Freiburg
**Leonid Libkin**§ Bell Laboratories
**Luc Segoufin**¶ INRIA

## Abstract

It is known that standard query languages for constraint databases lack the power to express connectivity properties. Such properties are important in the context of geographical databases, where one naturally wishes to ask queries about connectivity (what are the connected components of a given set?) or reachability (is there a path from A to B that lies entirely in a given region?). No existing constraint query languages that allow closed form evaluation can express these properties.

In the first part of the paper, we show that in principle there is no obstacle to getting closed languages that can express connectivity and reachability queries. In fact, we show that adding any topological property to standard languages like FO+Lin and FO+Poly results in a closed language. In the second part of the paper, we look for tractable closed languages for expressing reachability and connectivity queries. We introduce *path logic*, which allows one to state properties of paths with respect to given regions. We show that it is closed, has polynomial time data complexity for linear and polynomial constraints, and can express a large number of reachability properties beyond simple connectivity. Query evaluation in the logic involves obtaining a discrete abstraction of a continuous path, and model-checking of temporal formulae on the discrete structure.

## 1 Introduction

Several recent data models generalize the relational model by allowing direct modeling of structured database objects beyond the traditional flat tuple. Examples of the additional structure that can be modeled include nesting of tuples within other tuples, and the modeling of pointers and other datatypes in the object-oriented database model. We will deal in this paper with another such extension, the *constraint database model* [19, 23], in which database relations need not be simple finite collections of tuples, but can instead be constraint-definable collections, finite or infinite. The constraint model is appropriate for a variety of domains in which application data is naturally represented as solutions to constraints, such as geographic data and temporal data. Constraint databases allow queries to symbolically manipulate infinite collections of data, using both relational operators and the algebraic operations appropriate to the application domain.

We refer to the constraint model as the *polynomial constraint model* or the *linear constraint model*, depending on whether database objects are represented using general polynomial constraints or only linear constraints. More generally, we can parameterize the constraint model by any first-order structure $\mathcal{M}$ admitting a quantifier-elimination algorithm. In this general setting the 'constraint sets' that define database relations are simply the quantifier-free definable sets of $\mathcal{M}$.

Relational Calculus generalizes in an elegant and simple way to the constraint model. The analogous query languages use first-order logic (FO) over the vocabulary consisting of the schema relations supplemented with the operations of $\mathcal{M}$ (e.g., addition, multiplication). The implementation of the calculus reduces to

*Part of this work was done while M. Benedikt, M. Grohe and L. Libkin visited INRIA-Rocquencourt.

†Bell Laboratories, 263 Shuman Blvd, Naperville, IL 60566, USA. E-mail: benedikt@research.bell-labs.com.

‡Institut für Mathematische Logik, Eckerstr. 1, 79104 Freiburg, Germany. E-mail: grohe@logik.mathematik.uni-freiburg.de.

§Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA. E-mail: libkin@research.bell-labs.com.

¶INRIA-Rocquencourt, B.P. 105, Le Chesnay Cedex 78153, France. E-mail: Luc.Segoufin@inria.fr.

constraint-solving, or in the general case, quantifier-elimination. Many of the results and techniques of classical relational calculus, including complexity and expressivity bounds, carry over to these first-order constraint query languages [23]. However, first-order constraint queries are limited in much the same way that Relational Calculus is limited. Natural recursive queries, such as the transitive closure of a graph, remain inexpressible in first-order constraint query languages. Even more importantly, geometric analogs of these queries that are crucial for spatial database applications are inexpressible as well. The query

$$\texttt{CONNECTED}(S)$$

that tells whether a database relation $S$ is topologically connected, is inexpressible. The query

$$\texttt{CONNECTS-TO}(x, y, S)$$

that tells whether there is a path from point $x$ to point $y$ within relation $S$ is also inexpressible. Both of these results follow from [4]. The inability to express connectedness is a crucial obstacle in applying these languages to geographic databases. Although connectivity is perhaps the most natural geometric property that is absent from first-order constraint query languages, there are many other geometric properties which are conceptually (and even algorithmically) simple that are inexpressible as well: the query asking whether a planar region is simply connected, to take just one example. In fact, results of [20, 22] show, roughly, that the only purely topological facts of a single region expressible in first-order queries are 'local' – they merely assert the existence or nonexistence of points with a given topological type.

In this paper, we consider extensions of first-order constraint query languages that can express the reachability queries mentioned above, as well as other important non-local topological properties. Of course, in the context of the *relational* model, it is fairly well-understood how to add graph reachability as well as other tractable recursive queries to a first-order language; there are numerous results mapping out the query languages that result from such extensions, cf. [1]. What are the difficulties in extending beyond first-order logic in the constraint context?

When designing a query language one faces two major difficulties: achieving *closure* and *tractability*. A key idea behind constraint query languages is *closed-form evaluation*: if we start with databases definable over some structure $\mathcal{M}$ and apply an FO query, the result is again definable. In other words, the solution of a set of inequalities can be represented again as a set of inequalities. This closure property enables the use of a variety of inductive query-evaluation algorithms. We also want languages that are computationally *tractable*. The usual first-order constraint query languages have polynomial data complexity, as do standard languages for recursive querying of traditional relational databases.

Adding a traditional relational recursion mechanisms such as fixed points or while loops to first-order constraint query languages does *not* give this closure property, see [19]. In fact, the interaction of arithmetic with recursion can produce output databases that are not even computable, much less definable with constraints. In fact, queries for topological connectivity have been handled only through query languages that are both non-closed and computationally intractable [21, 12].

In contrast, our first main result shows that there is a way to add reachability and a vast number of other topological queries while retaining closure. In fact, for any set $\mathbb{T}$ of topological properties, we present a language, $\mathrm{FO}+\mathbb{T}$ that can define $\mathbb{T}$, and that is still closed. Moreover we can prove that the complexity of evaluating queries in $\mathrm{FO}+\mathbb{T}$ is polynomial in the complexity of checking the properties from $\mathbb{T}$. This implies a PTIME complexity for the extension of `CONNECTED`. The language $\mathrm{FO}+\mathbb{T}$ demonstrates that the closure problem by itself is no obstacle to admitting spatial reachability queries into constraint query languages.

Our second main result identifies a powerful logic, *Path Logic* – denoted $L_{\mathrm{PATH}}$ – which can define `CONNECTED`, `CONNECTS-TO` and other reachability queries, and which also admits effective, tractable query evaluation over linear and polynomial constraint databases. Path Logic has syntax and semantics that are reminiscent of traditional temporal logics used in the verification of reactive programs. Not only can query evaluation be done in polynomial data complexity, but we show that query evaluation reduces to a combination of *cell-decomposition algorithms* from real analytic geometry, followed by *model-checking* of a discrete system. In the model-checking phase, techniques from verification of discrete transition systems can be applied.

We will show positive results about both the tractability and expressivity of Path Logic, making the case that it is sufficiently expressive to capture the recursive queries that are most important to spatial database applications. Path Logic can thus be seen as a constraint database language that generalizes many recursive extensions of first-order logic, as well as a spatial analog of temporal logics such as CTL and CTL*.

**Related work** Most work on Datalog extensions for the constraint model deal with highly restrictive classes of constraints over integers [29], as over linear and

polynomial constraints Datalog is not closed. Topological connectivity for 2-dimensional polynomial constraint databases of degree 2 was shown to be definable in a language called Spatial Datalog [21]; later [12] extended this to arbitrary polynomial constraint databases. However, Spatial Datalog is likely to be computationally complete, and it certainly does not admit efficient evaluation. Results in computational algebraic geometry show that many connectivity and reachability queries (e.g., find the connected components of a set) are in PTIME if the dimension is fixed (see, e.g., [16]). However, languages capturing complexity classes over constraint databases are scarce (natural languages exist for databases definable with order constraints only [14]; also, a rather complicated language was given for linear constraints [15]). Besides, this approach can only work for those queries applied as top level operators (i.e., outputs cannot be reused by other queries). There exists extensive literature on first-order definable topological properties of constraint databases [26, 20, 22, 30] and it is well known that connectivity and reachability are not first-order [4]. Our results on application to hybrid systems are directly inspired by [24, 25].

**Organization** We introduce notations in Section 2. In Section 3, we deal with closure under topological properties. We first prove a number of general decomposition results for sets definable in various structures. We then use them to show closure of FO+Lin and FO+Poly under adding tests for topological properties, and tractability, assuming topological properties can be tested in polynomial time. We also study the special case of finding connected components.

In Section 4, we introduce Path Logic $L_{\text{PATH}}$ for expressing reachability and connectivity queries in a unified framework. We give examples, analyze expressive power, and prove closure. For linear and polynomial constraints we show tractability as well. We conclude by an application to verification of hybrid systems. All proofs can be found in the full version [5].

## 2 Notations

**Structures, databases, queries** Most notations are fairly standard in the literature on constraint databases, cf. [4, 6, 23, 27]. Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an infinite structure, where $\mathcal{U}$ is an infinite set, called a universe (in the database literature often called the domain), and $\Omega$ is a set of interpreted functions, constants, and predicates. In the field of constraint databases, most examples have $\mathcal{U} = \mathbb{R}$, the set of real numbers. Examples of signatures (and corresponding classes of constraints) that have been considered are:

| | |
|---|---|
| Dense Order Constraints: | $\langle \mathbb{R}, < \rangle$; |
| Linear Constraints: | $\mathbf{R}_{\text{lin}} = \langle \mathbb{R}, +, -, 0, 1, < \rangle$; |
| Polynomial Constraints: | $\mathbf{R} = \langle \mathbb{R}, +, \cdot, 0, 1, < \rangle$. |

A (relational) *database schema* $SC$ is a nonempty collection of relation names $\{S_1, \ldots, S_l\}$ with associated arities $p_1, \ldots, p_l > 0$. We shall consider *finitely representable*, or *definable* instances. A definable database instance of $SC$ over $\mathcal{M}$ is a family of sets $\{X_1, \ldots, X_l\}$, with $X_i \subseteq \mathcal{U}^{p_i}$, such that for each $X_i$ there exists a formula $\alpha_i(x_1, \ldots, x_{p_i})$ in the language of $\mathcal{M}$ with $X_i = \{\vec{a} \in \mathcal{U}^{p_i} \mid \mathcal{M} \models \alpha_i(\vec{a})\}$. Most applications of constraint databases consider definable instances over $\mathbf{R}_{\text{lin}}$ (called *semi-linear sets*) or over $\mathbf{R}$ (called *semi-algebraic sets*). These are sets definable by Boolean combinations of linear (resp., polynomial) inequalities.

As our basic query language, we consider relational calculus, or *first-order logic*, FO, over the underlying structure and the database schema. We use te notation $\text{FO} + \Omega$ to denote the class of all first-order formulae built up from the atomic $SC$ and $\Omega$ formulae by using Boolean connectives $\vee, \wedge, \neg$ and quantifiers $\forall, \exists$. When $\Omega$ is $(+, -, 0, 1, <)$, we use the notation FO + Lin (*first-order with linear constraints*), and when $\Omega$ is $(+, \cdot, 0, 1, <)$, we denote the language by FO + Poly (*first-order with polynomial constraints*).

Given $\varphi(\vec{x}, \vec{y})$ and $\vec{a}$, we write $\varphi(\vec{a}, D)$ for $\{\vec{b} \mid D \models \varphi(\vec{a}, \vec{b})\}$; in the absence of $\vec{x}$ we just write $\varphi(D)$ for the output of $\varphi$ on $D$. We say that a language $\text{FO} + \Omega$ is *closed*, if for any schema $SC$, any definable $SC$-database (over $\langle \mathcal{U}, \Omega \rangle$) and every $\text{FO} + \Omega$ query $\varphi(\vec{y})$ on $SC$-databases, the output $\varphi(D)$ is a definable set.

Languages FO+Lin, FO+Poly, as well as FO with dense order constraints are closed; this is a consequence of quantifier-elimination for $\mathbf{R}_{\text{lin}}$, $\mathbf{R}$ and $\langle \mathbb{R}, < \rangle$.

**O-minimality, cell decomposition** Many results that we prove extend beyond linear and polynomial constraints. To state them in greater generality, we use *o-minimality* [31], which plays an important role in the study of constraint query languages (cf. [4, 6, 23]).

A structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ is o-minimal, if every definable subset of $\mathcal{U}$ is a finite union of points and open intervals $(a, b) = \{x \mid a < x < b\}$, $(-\infty, a) = \{x \mid x < a\}$, and $(a, \infty) = \{x \mid x > a\}$ (we assume that $<$ is in $\Omega$). All the structures on the reals we mentioned so far – $\mathbf{R}_{\text{lin}}$, $\mathbf{R}$, $\langle \mathbb{R}, < \rangle$ – are o-minimal (this is implied by quantifier elimination and the fundamental theorem of algebra, for the case of $\mathbf{R}$.) There are a number of known o-minimal expansions of $\mathbf{R}$, most notably, the exponential

field $\langle \mathbb{R}, +, \cdot, e^x \rangle$ [32].

A key property of o-minimal structures is *cell decomposition*. A *cell* in $\mathbb{R}^k$ is a subset homeomorphic to $\mathbb{R}^{k'}$, $k' \leq k$ (by convention, $\mathbb{R}^0$ is a point). We now fix a structure $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ and define $\mathcal{M}$-cells by induction on dimension. An $\mathcal{M}$-cell in $\mathbb{R}^0$ is just $\mathbb{R}^0$. $\mathcal{M}$-cells in $\mathbb{R}$ are singletons $\{a\}$, or open intervals $(a,b), (-\infty, a), (a, \infty)$, where $a, b$ are definable constants. Assume that $C \subseteq \mathbb{R}^{n-1}$ is a cell, and $f, g : C \to \mathbb{R}$ are continuous definable function on $C$, with $f(x) < g(x)$ for all $x \in C$. Then the sets $\{(x, f(x)) \mid x \in C\}$ and $\{(x, r) \mid x \in C, f(x) < r < g(x)\}$ are cells in $\mathbb{R}^n$. In the latter case, we allow $f$ to be $-\infty$ and/or $g$ to be $\infty$.

A cell decomposition of $\mathbb{R}^n$ (with respect to $\mathcal{M}$) is a partition of $\mathbb{R}^n$ into a finite union of $\mathcal{M}$-cells. Again, it is defined inductively on $n$. A decomposition of $\mathbb{R}$ is the collection of $\mathcal{M}$-cells of the form $\{(-\infty, a_1), \{a_1\}, (a_1, a_2), \{a_2\}, \ldots, \{a_k\}, (a_k, \infty)\}$. A decomposition of $\mathbb{R}^n$ is a family $\mathcal{C} = \{C_1, \ldots, C_l\}$ of $\mathcal{M}$-cells that partition $\mathbb{R}^n$ such that for the natural projection $\pi : \mathbb{R}^n \to \mathbb{R}^{n-1}$ given by $\pi(\vec{x}, t) = \vec{x}$ for $\vec{x} \in \mathbb{R}^n, t \in \mathbb{R}$, the collection $\pi(\mathcal{C}) = \{\pi(C_1), \ldots, \pi(C_k)\}$ is a decomposition of $\mathbb{R}^{n-1}$.

In particular, if $\mathcal{C}$ is a decomposition of $\mathbb{R}^{n+m}$ and $\pi : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is the projection on the first $n$ coordinates, then $\pi(\mathcal{C})$ is a decomposition of $\mathbb{R}^n$, and for every cell $C \in \mathcal{C}$ there exists a unique cell $C_0 \in \pi(\mathcal{C})$ such that $C \subseteq C_0 \times \mathbb{R}^m$. This property, especially in the context of the real field, is referred to as being a *cylindric* decomposition.

**Fact 1** [31] *Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ be o-minimal. Assume that $S_1, \ldots, S_m$ are definable sets in $\mathbb{R}^n$. Then there exists a cell decomposition $\mathcal{C}$ of $\mathbb{R}^n$ such that each $S_i$ is a union of some cells of $\mathcal{C}$.* □

Let $X \subseteq \mathbb{R}^{n+m}$ and $\vec{a} \in \mathbb{R}^n$. Then $X_{\vec{a}}$ denotes the *fiber* $\{\vec{b} \in \mathbb{R}^m \mid (\vec{a}, \vec{b}) \in X\}$. Let $\mathcal{C}$ be a decomposition of $\mathbb{R}^{n+m}$ and $\vec{a} \in \mathbb{R}^n$. By $\mathcal{C}_{\vec{a}}$ we denote the collection of all nonempty sets $\{C_{\vec{a}} \mid C \in \mathcal{C}\}$. It is known [31] that $\mathcal{C}_{\vec{a}}$ is a cell decomposition of $\mathbb{R}^m$.

In the case when $\mathcal{M}$ is $\mathbf{R}_{\text{lin}}$ or $\mathbf{R}$, we can get more information about cell decompositions. Namely, given any finite collection $f_1(\vec{x}), \ldots, f_k(\vec{x})$ of polynomials (resp., linear functions) in $n$ variables with coefficients from $\mathbb{Q}$, one can find a cell decomposition $\mathcal{C}$ of $\mathbb{R}^n$ such that on each cell $C_i$, none of the functions $f_j$ changes its sign. Furthermore, this decomposition can be found in time $O((kd)^{h(n)})$, where $d$ is the maximal degree of a polynomial among $f_j$s, and $h$ is some function (typically, $h(n) = O(2^n)$) [10, 9]. It is important to notice that for a fixed dimension, the cell decomposition algorithm is thus in PTIME (in fact, in NC [7]).

## 3 Topological properties and closure

The goal of this section is to show that adding topological properties to languages like FO+Lin and FO+Poly results in *closed* query languages. In particular, one can add the connectivity test, or an operator computing connected components of a set, and still remain within semi-linear or semi-algebraic databases.

In Subsection 3.1, we prove a general decomposition result for definable sets that is key to the closure theorems of this section and next. Our starting point is the Local Triviality Theorem in real algebraic geometry, which implies, for example, that for a semi-algebraic set $S \subseteq \mathbb{R}^{n+m}$, the number of topological types of sets $S_{\vec{a}} \subseteq \mathbb{R}^m$ is finite, as $\vec{a}$ ranges over $\mathbb{R}^n$ [3, 8]. For $\mathbf{R}$, the known Local Triviality theorem gives the decompositions necessary for our results. In $\mathbf{R}_{\text{lin}}$, however, the Local Triviality Theorem fails, so we prove a weakening of it that is sufficient for the needs of the paper.

Once the decomposition lemma is proved, the general closure result for adding tests of topological properties follows easily. We treat it in Subsection 3.2, and then in Subsection 3.3 analyze more general topological operators, in particular, one for computing connected components of a set. We derive closure for FO+Poly and FO+Lin, although in very different ways: for FO+Poly it is an easy consequence of Local Triviality for semi-algebraic functions, while for FO+Lin the proof relies on the special form of decompositions.

### 3.1 Decomposition Lemma

We need a notion of decomposition into cells that is more general than the notion introduced in Section 2. Recall that a cell is a definable set homeomorphic to $\mathbb{R}^p$ for some $p \geq 0$. By a *decomposition* of $\mathbb{R}^k$ we mean a finite partition of $\mathbb{R}^k$ into cells $A_1, \ldots, A_l$. We consider special kinds of decompositions *of* $\mathbb{R}^{n+m}$ *over* $\mathbb{R}^n$. A decomposition $\mathcal{C}$ is *cylindric* over $\mathbb{R}^n$ if for every $A_i, A_j \in \mathcal{C}$, $\pi(A_i)$ and $\pi(A_j)$ are cells and either $\pi(A_i) = \pi(A_j)$, or $\pi(A_i) \cap \pi(A_j) = \emptyset$. Here $\pi$ is the projection $\mathbb{R}^{n+m} \to \mathbb{R}^n$. Note that the nonempty sets among $\pi(A_i)$s induce a decomposition on $\mathbb{R}^n$ which we denote by $\pi(\mathcal{C})$.

A decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$ is called *trivial* over $\mathbb{R}^n$ if it is cylindric over $\mathbb{R}^n$ and for any cell $C'$ of the induced decomposition $\pi(\mathcal{C})$ of $\mathbb{R}^n$, and for any $\vec{a}, \vec{b} \in C'$,

there exists a homeomorphism $h : \mathbb{R}^m \to \mathbb{R}^m$ such that $h(C_{\vec{a}}) = C_{\vec{b}}$ for every cell $C \in \mathcal{C}$.

Let $\mathrm{cl}(\cdot)$ denote the closure of a set (in the usual topology of $\mathbb{R}$). A decomposition $\mathcal{C}$ is called *adjacency preserving* over $\mathbb{R}^n$ if it is cylindric over $\mathbb{R}^n$ and for any cell $C'$ of the induced decomposition $\pi(\mathcal{C})$, for any $\vec{a}, \vec{b} \in C'$, and for all cells $A^1, A^2, A^3 \in \mathcal{C}$, $\mathrm{cl}(A^1_{\vec{a}}) \cap A^2_{\vec{a}} \neq \emptyset$ iff $\mathrm{cl}(A^1_{\vec{b}}) \cap A^2_{\vec{b}} \neq \emptyset$ and $\mathrm{cl}(A^1_{\vec{a}}) \cap \mathrm{cl}(A^2_{\vec{a}}) \cap A^3_{\vec{a}} \neq \emptyset$ iff $\mathrm{cl}(A^1_{\vec{b}}) \cap \mathrm{cl}(A^2_{\vec{b}}) \cap A^3_{\vec{b}} \neq \emptyset$. Note that a trivial decomposition is adjacency preserving.

**Lemma 2** *a) Let $S_1, \ldots, S_k$ be a collection of semi-linear sets in $\mathbb{R}^{n+m}$ and let $f_1, \ldots, f_p$ be all the (degree 1) polynomials used in the representation of $S_1, \ldots, S_k$. Then there exists a decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$ into semi-linear sets which is trivial over $\mathbb{R}^n$, and such that the sign of each $f_i$ is constant on every cell of $\mathcal{C}$. In particular, every $S_j$ is a union of cells of $\mathcal{C}$. Moreover, for $n$ and $m$ fixed, $\mathcal{C}$ can be found in time polynomial in the size of the descriptions of $f_1, \ldots, f_p$.*

*b) The statement a) holds if one replaces semi-linear with semi-algebraic.*

*c) Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ be o-minimal, and let $S_1, \ldots, S_k$ be a collection of definable sets in $\mathbb{R}^{n+m}$. Then there exists a decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$ which is adjacency preserving over $\mathbb{R}^n$ such that each set $S_i$ is a union of cells of $\mathcal{C}$. Moreover, if $\mathcal{M}$ is decidable, then $\mathcal{C}$ can be effectively computed.*

*d) If $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ is an o-minimal expansion of the real field $\mathbf{R}$, and $S_1, \ldots, S_k$ is a collection of definable sets in $\mathbb{R}^{n+m}$, then there exists a decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$ which is trivial and such that each set $S_i$ is a union of cells of $\mathcal{C}$.*

*Proof sketch.* The proof of b) and d) is by Local Triviality for the real field [3, 8] and o-minimal expansions [31]. As it fails over $\mathbf{R}_{\mathrm{lin}}$, the proof of a) is by combining standard cell decomposition techniques with results from convex analysis on the topology of polyhedra. The proof of c) is by cell decomposition and uniform bounds of [28]. □

### 3.2 Closure theorem

We now prove the closure result for topological properties. Formally, a *topological property* $\mathsf{Top}$ is a collection $\{\mathcal{T}_1, \ldots, \mathcal{T}_n, \ldots\}$ where $\mathcal{T}_n$ is a family of sets in $\mathbb{R}^n$ such that if $X \in \mathcal{T}_n$, then for each homeomorphism $h$ of $\mathbb{R}^n$, $h(X) \in \mathcal{T}_n$. For example, $\mathsf{Top}$ could express the property of being connected, being closed, being of dimension $n-1$, containing exactly one hole, etc. When the dimension $n$ is clear from the context, we write $X \in \mathsf{Top}$ instead of $X \in \mathcal{T}_n$.

For a set $\mathbb{T}$ of topological properties, we define the language $\mathrm{FO}(\Omega) + \mathbb{T}$ by extending the definition of $\mathrm{FO}(\Omega)$ with the following rule: if $\varphi(\vec{x}, \vec{y})$ is a query, then $\psi(\vec{x}) \equiv \mathsf{Top}\, \vec{y}.\ \varphi(\vec{x}, \vec{y})$ is a query. The semantic is as follows: $D \models \psi(\vec{a})$ iff $\varphi(\vec{a}, D) \in \mathsf{Top}$. Recall that $\varphi(\vec{a}, D) = \{\vec{b} \mid D \models \varphi(\vec{a}, \vec{b})\}$. For $\Omega$ being $(+, -, 0, 1, <)$ or $(+, \cdot, 0, 1, <)$ we use the notation $\mathrm{FO} + \mathrm{Lin} + \mathbb{T}$ and $\mathrm{FO} + \mathrm{Poly} + \mathbb{T}$.

For instance, the query "is the intersection of regions $R$ and $S$ connected" could be written as $\mathsf{C}\vec{x}.\ R(\vec{x}) \wedge S(\vec{x})$ (where we denote the property of being connected by $\mathsf{C}$). To illustrate the use of free variables, consider a set $S \subseteq \mathbb{R}^3$. Then the query $\varphi(x) \equiv \mathsf{C}(y, z).\ S(x, y, z)$ returns the set of all $c \in \mathbb{R}$ for which the intersection of $S$ with the plane $x = c$ is a connected set.

We say that the data complexity of $\mathrm{FO}(\Omega) + \mathbb{T}$ is $\mathrm{PTIME}^{\mathbb{T}}$ if $\mathrm{FO}(\Omega) + \mathbb{T}$ queries can be evaluated in polynomial time in the size of the database, assuming an oracle that can test each $\mathsf{Top} \in \mathbb{T}$ in constant time.

**Theorem 3** *Let $\mathbb{T}$ be any set of topological properties. Then $\mathrm{FO} + \mathrm{Lin} + \mathbb{T}$, $\mathrm{FO} + \mathrm{Poly} + \mathbb{T}$ and $\mathrm{FO}(\Omega) + \mathbb{T}$ are closed, for $\langle \mathbb{R}, \Omega \rangle$ an o-minimal expansion of the real field. Furthermore, the data complexity of $\mathrm{FO} + \mathrm{Lin} + \mathbb{T}$ and $\mathrm{FO} + \mathrm{Poly} + \mathbb{T}$ is $\mathrm{PTIME}^{\mathbb{T}}$.*

*Proof sketch.* Closure is proved by induction on the formulae. Consider $\psi(\vec{x}) \equiv \mathsf{Top}\, \vec{y}.\ \varphi(\vec{x}, \vec{y})$ for $\mathsf{Top} \in \mathbb{T}$. Let $\vec{x}$ and $\vec{y}$ be of length $n$ and $m$, resp., and let $D$ be a definable database. By induction, $\varphi(D)$ is a definable set $S \subseteq \mathbb{R}^n \times \mathbb{R}^m$. By Lemma 2, there is a decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$ trivial over $\mathbb{R}^n$ such that $S$ is a union of cells of $\mathcal{C}$. Let $\mathcal{C}'$ be the projection of $\mathcal{C}$ onto $\mathbb{R}^n$, and $C$ a cell in $\mathcal{C}'$. By triviality, for every $\vec{a}, \vec{b} \in C$, it is the case that $S_{\vec{a}}$ and $S_{\vec{b}}$ are homeomorphic, and thus they agree on $\mathsf{Top}$. Therefore, the output of $\psi$ on $D$ is a union of some cells in $\mathcal{C}'$ and thus definable.

In order to compute the output of $\psi$ on $D$, it suffices to test, for each cell $C$ in $\mathcal{C}'$, whether it belongs to the output or not. By triviality, this can be done by checking for a sample point $\vec{a} \in C$, whether $S_{\vec{a}} \in \mathsf{Top}$. By Lemma 2, there are polynomially many cells $C$, which implies the complexity claimed in the Theorem. □

Now consider the case when $\mathbb{T}$ consists of just the property $\mathsf{C}$ (being a connected set). As connectivity of semi-algebraic sets can be tested in polynomial time (for a fixed dimension) [16], the proof of the complexity bounds in Theorem 3 implies the following.

**Corollary 4** FO + LIN + C *and* FO + POLY + C *are closed, and the queries they define have* PTIME *data complexity.* □

### 3.3 Topological queries and connected components

So far we have seen closure and tractability for languages which add the CONNECTED operator mentioned in the introduction. We now deal with the CONNECTS-TO operator; that is, with computing connected components. In fact, we treat a more general case of non-boolean topological queries in the context of polynomial constraints.

Let $T$ be a map from subsets of $\mathbb{R}^m$ to subsets of in $\mathbb{R}^{mk}$. We call $T$ *topological* if for any homeomorphism $h : \mathbb{R}^m \to \mathbb{R}^m$ and any $(\vec{x}_1, \ldots, \vec{x}_k) \in T(S)$, for $S \subseteq \mathbb{R}^m$, we have $(h(\vec{x}_1), \ldots, h(\vec{x}_k)) \in T(h(S))$. For example, the mapping $\mathsf{Conn} : 2^{\mathbb{R}^m} \to 2^{\mathbb{R}^m \times \mathbb{R}^m}$ such that $(\vec{x}_1, \vec{x}_2) \in \mathsf{Conn}(S)$ iff $\vec{x}_1, \vec{x}_2$ are in the same connected component of $S$, is topological. We say that $T$ is *definable* if $T(S)$ is a definable set for every definable set $S$. As connected components of a set definable in an o-minimal structure on $\mathbb{R}$ are definable [31], the topological query $\mathsf{Conn}$ is definable over such structures.

Let $T : 2^{\mathbb{R}^m} \to 2^{\mathbb{R}^{mk}}$ be a topological query. We define the language $\mathrm{FO}(\Omega) + T$ by extending the definition of $\mathrm{FO}(\Omega)$ with the following rule: if $\varphi(\vec{x}, \vec{y})$ is a query with $\vec{y}$ having length $m$, then we get a new query $\psi(\vec{x}, \vec{y}_1, \ldots, \vec{y}_k) \equiv T\vec{y}.\ \varphi(\vec{x}, \vec{y})$, with all $\vec{y}_i$s having length $m$. The semantic is as follows:

$$D \models \psi(\vec{a}, \vec{b}_1, \ldots, \vec{b}_k) \text{ iff } (\vec{b}_1, \ldots, \vec{b}_k) \in T(\varphi(\vec{a}, D)).$$

For example, if $\varphi(D)$, for $\varphi(\vec{x}, \vec{y})$, is a set $S \subseteq \mathbb{R}^{n+m}$, and $\psi(\vec{x}, \vec{y}_1, \vec{y}_2) \equiv \mathsf{Conn}\vec{y}.\ \varphi$, then $D \models \psi(\vec{a}, \vec{b}_1, \vec{b}_2)$ iff $\vec{b}_1$ and $\vec{b}_2$ are in the same connected component of $S_{\vec{a}}$.

In the semi-algebraic case or in any o-minimal expansion, the Local Triviality Theorem used in the proof of Lemma 2 can also be used to prove the following:

**Theorem 5** *For every definable topological map $T$,* $\mathrm{FO} + \mathrm{POLY} + T$ *is closed. Moreover,* $\mathrm{FO}(\Omega) + T$ *is closed, whenever $\langle \mathbb{R}, \Omega \rangle$ is an o-minimal expansion of the real field.*

*Proof sketch.* By induction on the structure of the formulae. Consider the case of $\psi(\vec{x}, \vec{y}_1, \ldots, \vec{y}_k) \equiv T\vec{y}.\ \varphi(\vec{x}, \vec{y})$. Assume that $\varphi$ defines $S \subseteq \mathbb{R}^n \times \mathbb{R}^m$. By Lemma 2, there exists a decomposition $\mathcal{C}$ trivial over $\mathbb{R}^n$. Let $C_1, \ldots, C_p$ be all the cells in the projection of $\mathcal{C}$ onto $\mathbb{R}^n$. The proof of b) in Lemma 2 implies that for every $i$, and every $\vec{a}, \vec{b} \in C_i$, there is a definable homeomorphism $h^i_{\vec{a},\vec{b}} : \mathbb{R}^m \to \mathbb{R}^m$ such that $h^i_{\vec{a},\vec{b}}(S_{\vec{a}}) = S_{\vec{b}}$. Hence, $(\vec{e}_1, \ldots, \vec{e}_k) \in T(S_{\vec{a}})$ iff $(h^i_{\vec{a},\vec{b}}(\vec{e}_1), \ldots, h^i_{\vec{a},\vec{b}}(\vec{e}_k)) \in T(S_{\vec{b}})$.

Since each cell $C_i$ is a definable set, it has a definable representative $\vec{c}_i \in C_i$ [31]. Thus, a tuple $(\vec{a}, \vec{b}_1, \ldots, \vec{b}_k)$ is in $\psi(D)$ iff for $i \in 1, \ldots, p$ such that $\vec{a}$ is in $C_i$, the following holds:

$$\exists (\vec{e}_1, \ldots, \vec{e}_k) \in T(S_{\vec{c}_i}) \ \bigwedge_{j=1}^{p} \vec{b}_j = h^i_{\vec{c}_i,\vec{a}}(\vec{e}_j).$$

This implies that $\psi(D)$ is a definable set, and proves closure. □

We note in passing that the fact that $T$ produces a definable output on a definable input by no means implies closure. For example, the convex hull operator preserves semi-linearity, but when added to FO+LIN, gives it the full power of FO+POLY [2]. One can find definable operators that, when added to FO+POLY, define non-semi-algebraic sets (e.g., given two sets $X$ and $Y$ in $\mathbb{R}^n$, return the one with the larger volume).

Since $\mathsf{Conn}$ is definable and topological (over the real field and its o-minimal expansions), we conclude from Theorem 5 that $\mathrm{FO} + \mathrm{POLY} + \mathsf{Conn}$ is closed. However, the proof above cannot possibly be extended to FO + LIN. Indeed, we used not only the triviality of the partition which is guaranteed by Lemma 2, but also the fact that homeomorphisms between fibers are definable. This latter condition fails over $\langle \mathbb{R}, +, -, 0, 1, < \rangle$. Nevertheless, we can show that $\mathrm{FO} + \mathrm{LIN} + \mathsf{Conn}$ is closed.

**Proposition 6** FO+LIN+Conn *is closed; that is, it defines a semi-linear output on a semi-linear input. Furthermore,* $\mathrm{FO} + \mathrm{LIN} + \mathsf{Conn}$ *queries have* PTIME *data complexity.*

*Proof sketch.* The proof is by induction on the formulae, with only the case of applying the $\mathsf{Conn}$ operator being nontrivial. Suppose we are given $\varphi(\vec{x}, \vec{y})$ which defines, on a semi-linear database $D$, a semi-linear set $S \subseteq \mathbb{R}^n \times \mathbb{R}^m$. We must show that the set $S' \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$ of triples $(\vec{a}, \vec{b}_2, \vec{b}_2)$ such that $\vec{b}_1$ and $\vec{b}_2$ are in the same connected component of $S_{\vec{a}}$, is semi-linear, and can be computed in time polynomial in the size of a given representation of $S$ (assuming $n$ and $m$ fixed). Using Lemma 2, compute in PTIME a decomposition $\mathcal{C}$ of $\mathbb{R}^{n+m}$, which is trivial over $\mathbb{R}^n$, such that the signs of all the functions used in the representation of $S$ remain constant on each cell.

Let $C$ be a cell in $\pi(\mathcal{C})$, where $\pi : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is the natural projection. Let $C^1, \ldots, C^k$ be all the cells in the

cylinder $C \times \mathbb{R}^m$. For any $\vec{a}, \vec{b}$, $C^i_{\vec{a}}$ adj $C^j_{\vec{a}}$ iff $C^i_{\vec{b}}$ adj $C^j_{\vec{b}}$ iff $C^i$ adj $C^j$ where $X$ adj $Y$ means $X \cap \mathrm{cl}(Y) \neq \emptyset$ or $Y \cap \mathrm{cl}(X) \neq \emptyset$. We let $G_C$ be a graph with the set of nodes being the indices of the cells among $C^1, \ldots, C^k$ that belong to $S$, and edges $(i,j)$ for every $C^i$ adj $C^j$. Let $K_1, \ldots, K_p$ be the connected components of $G_C$. It is known [31] that the sets of the form $(\bigcup_{i \in K_l} C^i)$ are exactly the connected components of $S \cap (C \times \mathbb{R}^m)$. The result cane now easily be derived from this. □

We note that the results on closure under topological operators and Conn leave something to be desired. First, the proof of Theorem 5 does not produce a complexity bound, as it is not immediately clear how hard it is to compute definable homeomorphisms between fibers. The proof for FO+LIN, although giving us tractability, is rather ad-hoc, and slight modification of a query may require an entirely different proof of closure.

We now want to find a single language that captures the properties that are of interest for applications, which has a small number of constructors, and which has a uniform evaluation method over all queries. Such a language is presented in the next section.

## 4 Path Logic

Our goal is to present a unifying query language for expressing reachability and connectivity queries. The language is based on the concept of a path and allows to express properties of paths with respect to given sets in the Euclidean space $\mathbb{R}^n$. For now, let us think of a path as a continuous curve in $\mathbb{R}^n$. For example, to express that a set $S \subseteq \mathbb{R}^n$ is connected we would say "for all $\vec{x}, \vec{y} \in S$ there exists a path such that for all points $p$ on this path which appear between $\vec{x}$ and $\vec{y}$ we have $p \in S$". Formally, we would write this in the form

$$\forall \vec{x} \forall \vec{y} \Big( \big(S(\vec{x}) \wedge S(\vec{y})\big) \longrightarrow \mathbf{EP}\, \exists p \exists q \big(p = \vec{x} \wedge q = \vec{y} \wedge \forall r((p < r \wedge r < q) \rightarrow S(r))\big)\Big).$$

Let us try to decode this formula: The first line is just first-order, its obvious meaning is "for all tuples $\vec{x}, \vec{y} \in \mathbb{R}^n$ which are both contained in $S$ we have." Then **EP** in the second line says "there exists a path," i.e. a continuous curve in $\mathbb{R}^n$. We shall assume that all paths have a starting point; that is, they are continuous maps $f : \mathbb{R}_+ \to \mathbb{R}^n$ where $\mathbb{R}_+ = \{r \in \mathbb{R} \mid r \geq 0\}$. Next we quantify over a new type of variables, *path variables* $p, q, r$, which range over the points of the path. We say that "there exist points $p, q$ on the path such that $p$ equals $\vec{x}$ and $q$ equals $\vec{y}$ (if we consider them as points of $\mathbb{R}^n$), and all points $r$ between $p$ and $q$ are contained in $S$." Here the order in "between" is just the natural order on $\mathbb{R}$. So formally the meaning of the second line of our formula is

$$\exists t, u \in \mathbb{R}_+ \big(f(t) = \bar{x} \wedge f(u) = \vec{y} \wedge \forall v \in (t,u)\, f(v) \in S\big).$$

Similarly, we can formulate statements "for all paths" by using a universal quantifier **AP** instead of **EP**.

Before we give the formal definition of the syntax and semantics of our logic, let us give one more intuitive example. The following query defines the set of all $\vec{y} \in \mathbb{R}^2$ such that if one wants to go from a point in Portugal ($P \subseteq \mathbb{R}^2$) to $\vec{y}$ on land ($L$) then one has to go through Spain ($S$) and then France ($F$):

$$\forall \vec{x} \Big( P(\vec{x}) \longrightarrow \mathbf{AP}\, \Big( \big(\forall p\, L(p) \wedge \exists p \exists q (p < q \wedge p = \vec{x} \wedge q = \vec{y})\big) \rightarrow \exists r \exists s \big(r < s \wedge S(r) \wedge F(s)\big)\Big)\Big).$$

Thus the query returns, for example, all points $\vec{y}$ in France, Germany, and Italy, but no points in Spain or England.

In Subsection 4.1, we formally define the path logic $L_{\mathrm{PATH}}$. In Subsection 4.2, we give more examples and analyze the expressive power. In Subsection 4.3, we show that $L_{\mathrm{PATH}}$ is closed and tractable over polynomial and linear constraints (more generally, the closure is shown for o-minimal structures). Finally, Subsection 4.4 gives an application to hybrid systems.

### 4.1 Definition of the path logic

Formulas in the logic may have two sorts of variables, *element variables* $x, y, \ldots$ and *path variables* $p, q \ldots$. There are also two kinds of formulae: *state formulae* and *path formulae*. Associated with each path formula is an *arity* $n \geq 1$. (An $n$-ary path formula speaks about a path in $\mathbb{R}^n$.)

**Syntax** Given a database schema $SC$ and a structure $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$, formulae of $L_{\mathrm{PATH}}(\Omega)$ are defined inductively as follows:

1. Every FO + $\Omega$ formula $\varphi$ is a state formula.
2. State formulae are closed under the Boolean connectives $\vee, \wedge, \neg$, and quantification $\exists, \forall$.
3. If $\varphi$ is a state formula, $\vec{x} = (x_1, \ldots, x_n)$ is an $n$-tuple of element variables, and $p$ is a path variable, then $\varphi[p \leftarrow \vec{x}]$ is a path formula of arity $n$.

4. If $n \geq 1$ and $p, q$ are path variables, then $p = q$, $p < q$, and $p > q$ are path formulas of arity $n$. (To be formally correct, we should write $p =_n q$, $p <_n q$, or $p >_n q$ for the $n$-ary versions of these formulae, but we can safely omit this.)

5. Path formulae are closed under the Boolean connectives $\vee, \wedge, \neg$.

6. If $\varphi$ is a path formula and $p$ a path variable, then $\exists p\, \varphi$ and $\forall p\, \varphi$ are path formulae of the same arity as $\varphi$.

7. If $\varphi$ is a path formula without free path variables, then $\mathbf{EP}\, \varphi$ and $\mathbf{AP}\, \varphi$ are state formulae.

To make the last point of the definition precise, we have to define the set $\mathrm{FV_p}(\psi)$ of free path variables of a formula $\psi$: If $\psi \equiv \varphi[p \leftarrow \vec{x}]$ then $\mathrm{FV_p}(\psi) = \{p\}$. For the other types of path formulas, $\mathrm{FV_p}$ is defined in the usual way, for example $\mathrm{FV_p}(p < q) = \{p, q\}$, $\mathrm{FV_p}(\varphi_1 \vee \varphi_2) = \mathrm{FV_p}(\varphi_1) \cup \mathrm{FV_p}(\varphi_2)$, and $\mathrm{FV_p}(\exists p\, \varphi) = \mathrm{FV_p}(\varphi) \setminus \{p\}$. If $\psi$ is a state formula, then $\mathrm{FV_p}(\psi) = \emptyset$.

This completes the definition of the syntax of $L_{\mathrm{PATH}}(\Omega)$. When $\Omega = (+, -, 0, 1, <)$, we use the notation $L_{\mathrm{PATH}}(\textsc{Lin})$ for $L_{\mathrm{PATH}}(\Omega)$; for $\Omega = (+, \cdot, 0, 1, <)$, we use the notation $L_{\mathrm{PATH}}(\textsc{Poly})$.

We shall usually write $p = \vec{y}$ instead of $\vec{x} = \vec{y}[p \leftarrow \vec{x}]$. Similarly, if $R$ is a relation name, then we shall write $R(p)$ instead of $R(\bar{x})[p \leftarrow \bar{x}]$. (We have already used these conventions in the examples at the beginning of this section.)

To be able to define the semantics, we also have to define the set $\mathrm{FV_e}(\psi)$ of free element variables of a formula $\psi$: If $\psi$ is an FO $+\,\Omega$-formula, then $\mathrm{FV_e}(\psi)$ is the set of free variables of $\psi$ defined in the usual way. Similarly, if $\psi$ is a Boolean combination of two state or path formulas, then we apply the usual rules to define $\mathrm{FV_e}(\psi)$. If $\psi \in \{\exists x\, \varphi, \forall x\, \varphi\}$ (for an element variable $x$) then $\mathrm{FV_e}(\psi) = \mathrm{FV_e}(\varphi) \setminus \{x\}$. If $\psi \in \{\exists p\, \varphi, \forall p\, \varphi\}$ (for a state variable $p$) then $\mathrm{FV_e}(\psi) = \mathrm{FV_e}(\varphi)$. If $\psi \in \{\mathbf{EP}\, \varphi, \mathbf{AP}\, \varphi\}$ then $\mathrm{FV_e}(\psi) = \mathrm{FV_e}(\varphi)$. Finally, if $\psi \equiv \varphi[p \leftarrow (x_1, \ldots, x_n)]$ then $\mathrm{FV_e}(\psi) = \mathrm{FV_e}(\varphi) \setminus \{x_1, \ldots, x_n\}$.

**Semantics** In our informal discussion starting this section we defined a path in $\mathbb{R}^n$ to be an arbitrary continuous mapping $P : \mathbb{R}_+ \to \mathbb{R}^n$. However, continuous functions can oscillate very wildly, and including pathological curves may lead to counterintuitive truth values for sentences. Therefore, we define our semantics with respect to the set of *non-zeno* paths, which are reasonably smooth and can only oscillate mildly. Formally, given an o-minimal structure $\mathcal{M}$, we say that a path $P : \mathbb{R} \to \mathbb{R}^n$ is non-zeno with respect to $\mathcal{M}$ if for any set $X \subseteq \mathbb{R}^n$ definable in any o-minimal expansion of $\mathcal{M}$, the set $P^{-1}(X) = \{t \mid P(t) \in X\}$ is a union (not necessarily finite) of intervals, and the set of endpoints of those intervals is discrete.

For example, every semi-algebraic path is non-zeno with respect to $\mathbf{R}_{\mathrm{lin}}$ and $\mathbf{R}$. Another typical class of non-zeno paths can be obtained as follows: Let $\mathbb{R}$ be partitioned into intervals $I_1, I_2, \ldots$ (which could be open or closed on each side) such that for some $\epsilon > 0$, the length of each $I_j$ is at least $\epsilon$. Let $P : \mathbb{R}_+ \to \mathbb{R}^n$ be piece-wise semi-algebraic with respect to this partition; that is, $P$ is continuous and its restriction to each $I_j$ is semi-algebraic. Then $P$ is non-zeno with respect to $\mathbf{R}_{\mathrm{lin}}$ and $\mathbf{R}$. An example is given by $P : \mathbb{R}_+ \to \mathbb{R}^2$ defined as $P(x) = (x, x - \lfloor x \rfloor)$ if $\lfloor x \rfloor$ is even, and $P(x) = (x, \lceil x \rceil - x)$ if $\lfloor x \rfloor$ is odd. An example of a path that is non-zeno with respect to $\mathbf{R}_{\mathrm{lin}}$ and $\mathbf{R}$, but not piecewise semi-algebraic, is the path $s : \mathbb{R} \to \mathbb{R}^2$ defined by $s(x) := (x, \sin(x))$.

We now give the formal definition of the semantics of $L_{\mathrm{PATH}}(\Omega)$. Whereas for state formulas the satisfaction relation is defined with respect to databases $D$ in the usual way, the satisfaction relation for path formulas is defined with respect to pairs $(D, P)$ consisting of a database $D$ and a non-zeno path $P$.

The cases of FO $+\,\Omega$ formulae, as well as first-order quantification and Boolean connectives are standard.

For an $n$-ary path formula $\psi(p, \vec{y}) \equiv \varphi(\vec{x}, \vec{y})[p \leftarrow \vec{x}]$, a database $D$, a non-zeno path $P : \mathbb{R}_+ \to \mathbb{R}^n$, a $t \in \mathbb{R}_+$, and a tuple $\vec{a} \in \mathbb{R}^m$ with $m = |\vec{y}|$, we have $(D, P) \models \psi(t, \vec{a})$ iff $D \models \varphi(P(t), \vec{a})$. For $\varphi(p, q) \equiv (p \theta q)$, with $\theta \in \{<, =, >\}$, we define $(D, P) \models \varphi(t, t')$ iff $t \theta t'$. Finally, $D \models \mathbf{EP}\, \varphi(\vec{a})$ iff there exists a non-zeno path $P$ such that $(D, P) \models \varphi(\vec{a})$.

This completes the definition of the semantics of $L_{\mathrm{PATH}}$.

We close this section with an example that shows why we have to be very careful in defining a path logic that is closed and decidable. Note that in formulae $\varphi[p \leftarrow \vec{x}]$, only one path variable can get instantiated. Intuitively, this corresponds to coloring a path with previously defined regions. The following example shows why one cannot bind two path variables at the same time, i.e. admit formulas of the form $\varphi[p \leftarrow \vec{x}, q \leftarrow \vec{y}]$ (with the obvious meaning).

**Example 7** Let $\alpha(x_1, x_2) \equiv (x_2 = 0)$. Then the path formula

$$\beta(p, q) \equiv \Big( \alpha[p \leftarrow \vec{x}] \wedge \alpha[q \leftarrow \vec{x}] \wedge p < q \\ \wedge \neg \exists r \big( \alpha[r \leftarrow \vec{x}] \wedge p < r \wedge r < q \big) \Big)$$

says that $p$ and $q$ are two consecutive intersections of a path with the $x_1$-axis. Now let

$$\gamma(x_1, x_2, y_1, y_2) \equiv (x_2 = 0) \wedge (y_2 = 0) \wedge (y_1 = x_1 + 1)$$

and consider the formula

$$\varphi(x) \equiv \mathbf{EP}\Big( \begin{array}{c} \exists p \exists q \big(p = (0,0) \wedge q = (0,x) \wedge p < q\big) \\ \wedge \forall p \forall q \big(\beta(p,q) \rightarrow \gamma[p \leftarrow \vec{x}, q \leftarrow \vec{y}]\big)\Big). \end{array}$$

It says that there is a path from $(0,0)$ to $(x,0)$ in $\mathbb{R}^2$ such that two consecutive points of the form $(x_1,0),(x_2,0)$ on this path must satisfy $x_2 = x_1 + 1$. Hence, $\varphi(x)$ holds iff $x$ is an integer, while $\mathbb{Z}$ is *not* a semi-algebraic set.

### 4.2 Expressive power

In this section we collect a few facts we know about the expressive power of $L_{\mathrm{PATH}}$. We have indicated that `CONNECTED` and `CONNECT-TO` can be expressed: for example, to test $\vec{x}, \vec{y}$ are in the same connected component of $R$, one writes $\big(R(\vec{x}) \wedge R(\vec{y})\big) \wedge \mathbf{EP}\,\big(\forall p\, R(p) \wedge \exists p_1(p_1 = \vec{x}) \wedge \exists p_2(p_2 = \vec{y})\big)$.

As another example, we show how to test if a region $R \subseteq \mathbb{R}^2$ is *simply connected*. Intuitively, $R$ being simply connected means there are no holes in it (formally, a connected region $R$ is simply connected if every closed curve in it is homotopic to a single point). Note that it is easy to check if a connected region $R$ has a hole: either $\mathbb{R}^2 \setminus R$ is bounded, or this is not the case, and there are points $\vec{x}, \vec{y}$ such that every path from $\vec{x}$ to $\vec{y}$ intersects $R$. Clearly, this can be expressed in $L_{\mathrm{PATH}}$.

The example above is an instance of a general result, saying that the language $L_{\mathrm{PATH}}$ is quite expressive when it comes to topological queries in 2-dimensional space. Proposition 8 below can be used to express many more 2-dimensional topological queries. Note that this is particularly relevant in geographical information systems, which most often deal with 2-dimensional data.

With every 2-dimensional spatial database instance $D$, one can associate a finite structure $I(D)$, its *topological invariant* [26]. $I(D)$ captures the topological information about $D$, which means that two instances $D$ and $D'$ are homeomorphic if, and only if, $I(D)$ and $I(D')$ are isomorphic. We now show the definability of $I(D)$ in $L_{\mathrm{PATH}}$.

**Proposition 8** *The topological invariant of semi-linear or semi-algebraic 2-dimensional spatial database instances is definable in* $L_{\mathrm{PATH}}$. *More precisely, the topological invariant of semi-linear instances is definable in* $L_{\mathrm{PATH}}(\textsc{Lin})$, *and the topological invariant of semi-algebraic instances is definable in* $L_{\mathrm{PATH}}(\textsc{Poly})$. □

In the following section, we will see that every $L_{\mathrm{PATH}}$-query is computable in polynomial time. We do not believe that the converse is true, although proving this remains open. $L_{\mathrm{PATH}}$ seems to be closer to NL (nondeterministic logarithmic space) than to PTIME. The following proposition may illustrate this:

**Proposition 9** *Every query on finite constraint databases that is computable in* NL *is expressible in* $L_{\mathrm{PATH}}$.

*Proof sketch.* By well-known results from descriptive complexity theory [18], to prove this proposition it suffices to define *directed reachability* in $L_{\mathrm{PATH}}$. This can be done by first defining a topological embedding of the input graph in $\mathbb{R}^3$ and then using suitable paths in this embedded graph. □

Of course this proposition is only a small step towards an understanding of the expressive power of $L_{\mathrm{PATH}}$. We would like to prove the converse statement that every $L_{\mathrm{PATH}}$-query on finite instances is computable in NL. Unfortunately, this seems to be quite difficult. One approach would be to prove a collapse result saying that generic $L_{\mathrm{PATH}}$-queries on finite structures are all expressible in some finitary path logic, and then use results from finite model theory. It is not clear how to extend first-order collapse results to $L_{\mathrm{PATH}}$, however.

Another difficult problem is to compare the expressive power of $L_{\mathrm{PATH}}$ with that of the various extensions of FO by the connectivity quantifiers. We believe that directed reachability in finite graphs is not expressible in FO+Poly+Conn, but this appears to be very hard to prove, even under the complexity theoretic assumption that undirected reachability is not NL-complete.

### 4.3 Query evaluation: Closure and complexity

While $L_{\mathrm{PATH}}$ can express a great deal of reachability queries in constraint databases, it is not immediately clear whether it is either closed or tractable. Indeed, we saw in Example 7 that if one extends $L_{\mathrm{PATH}}$ by allowing new binary predicates on path variables, the resulting logic is neither closed nor decidable. We now show that $L_{\mathrm{PATH}}$ has very desirable closure and tractability properties for domains relevant in spatial applications. More precisely, we say that $L_{\mathrm{PATH}}(\Omega)$

- is *decidable* if for every definable (in $\langle \mathbb{R}, \Omega \rangle$) database $D$, and every state sentence $\Phi$, it is decidable whether $D \models \Phi$;

- is *closed* if for every definable database and every state formula $\varphi(\vec{x})$, the set $\{\vec{a} \mid D \models \varphi(\vec{a})\}$ is definable;
- *admits effective query evaluation* (of data complexity $\mathcal{C}$) if in addition a formula defining the set $\{\vec{a} \mid D \models \varphi(\vec{a})\}$ can be effectively obtained (and the complexity of obtaining such a formula is $\mathcal{C}$ in terms of the size of the database $D$).

Our goal is to prove the following result.

**Theorem 10** *Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ be o-minimal and decidable. Then $L_{\mathrm{PATH}}(\Omega)$ is closed and decidable. Furthermore, if $\mathcal{M}$ admits effective quantifier-elimination, then $L_{\mathrm{PATH}}(\Omega)$ admits effective query evaluation, and for $L_{\mathrm{PATH}}(\mathrm{LIN})$ and $L_{\mathrm{PATH}}(\mathrm{POLY})$ the data complexity is* PTIME.

*Proof sketch.* The proof is rather involved and consists of two main steps. We first show that an $L_{\mathrm{PATH}}$ query can be transformed into a "discrete path query" in which instead of quantifying over any time in a path $P$, we quantify only over components of the path with respect to some discretization of the path. In the second step we show that these discrete path queries can be evaluated on a database by model-checking the adjacency structure of the appropriate cell-decomposition provided by Lemma 2. □

Note that the proofs in Subsection 3.3 established tractability of $\mathrm{FO} + \mathrm{LIN} + \mathsf{Conn}$ but not $\mathrm{FO} + \mathrm{POLY} + \mathsf{Conn}$. Since connected components are definable in $L_{\mathrm{PATH}}(\mathrm{POLY})$, we conclude now:

**Corollary 11** $\mathrm{FO} + \mathrm{POLY} + \mathsf{Conn}$ *queries have* PTIME *data complexity.* □

### 4.4 Model-checking for hybrid systems

We mentioned that our logic $L_{\mathrm{PATH}}$ has been inspired by temporal logics used in the model-checking approach to automated verification of finite-state reactive systems. Recent work on real-time and hybrid systems includes a number of formalisms for expressing reachability properties of infinite state systems defined from real parameters [17]. We now show that our results can be applied in this area. More precisely, we show that the *model-checking problem* for linear time temporal logic LTL for one class of hybrid systems, the o-minimal hybrid systems of [24, 25] is decidable and, furthermore, tractable, if the dimension of the hybrid systems is fixed. It is straightforward to extend our approach to the branching time temporal logics CTL and CTL$^*$.

A *hybrid system* (cf. [17, 24]) of dimension $n$ is a tuple $H = (S, S_0, S_F, F, E, I, G, R)$, where

- $S = Q \times \mathbb{R}^n$, where $Q$ is a finite set, is the *state space*,
- $S_0 \subseteq S$ is the set of *initial states*,
- $S_F \subseteq S$ is the set of *final states*,
- $F : S \to \mathbb{R}^n$ assigns to each $q \in Q$ a vector field $F(q, \cdot)$,
- $E \subseteq Q \times Q$ is a set of *discrete transitions*,
- $I : Q \to 2^{\mathbb{R}^n}$ assigns to each $q \in Q$ a set $I(q)$ called the *invariant of* $q$,
- $G : E \to 2^{\mathbb{R}^n}$ assigns to each discrete transition $e = (q_1, q_2) \in E$ a set $G(e) \subseteq I(q_1)$ called the *guard of* $e$,
- $R : E \to 2^{\mathbb{R}^n}$ assigns to each discrete transition $e = (q_1, q_2) \in E$ a set $R(e) \subseteq I(q_2)$ called the *reset of* $e$.

Associated with the hybrid system $H$ is a ternary *transition relation* $\to \subseteq S \times (E \cup \{c\}) \times S$, where $c$ is a new symbol not contained in $E$. We write $s \xrightarrow{e} s'$ instead of $(s, e, s') \in \to$. We have two kinds of transitions:

- *Discrete Transitions:* $(q, \vec{x}) \xrightarrow{e} (q', \vec{x}')$ iff $e = (q, q') \in E$ and $\vec{x} \in G(e)$, $\vec{x}' \in R(e)$.
- *Continuous Transitions:* $(q, \vec{x}) \xrightarrow{c} (q', \vec{x}')$ iff $q = q'$ and there exists a $\delta \geq 0$ and a curve $x : [0, \delta] \to \mathbb{R}^n$ such that $x(0) = \vec{x}$, $x(\delta) = \vec{x}'$ and for every $t \in [0, \delta]$ it satisfies $\dot{x}(t) = F(q, x(t))$ and $x(t) \in I(q)$.

We assume that our hybrid systems are *non-blocking*, that is, for every state $s \in S$ there is an $e \in E \cup \{c\}$ and a state $s'$ such that $s \xrightarrow{e} s'$.

A *trajectory* of $H$ is a sequence $s_1 e_i s_2 e_2 \ldots$ such that for all $i \geq 1$ we have $s_i \xrightarrow{e_i} s_{i+1}$.

An *interpreted hybrid system* of signature $\Sigma = \{\pi_1, \ldots, \pi_m\}$ consists of a hybrid system $H$ and a mapping $\Pi$ that assigns to each state $s \in S$ a subset of $\Sigma$. Then $\Pi$ associates with each trajectory $\tau = s_1 e_1 s_2 e_2 \ldots$ of $H$ an $\omega$-word $\Pi(\tau) := \Pi(s_1)\Pi(s_2)\ldots$ over the alphabet $2^\Sigma$. We assume that the reader is familiar with the linear time temporal logic LTL (interpreted over $\omega$-words), see [11]. The LTL-*model checking problem for hybrid systems* is defined as follows:

*Input:* An interpreted hybrid system $(H, \Pi)$ and an LTL-formula $\varphi$.

*Problem:* Decide if for every trajectory $\tau$ of $H$ the word $\Pi(\tau)$ satisfies $\varphi$.

Let $\mathcal{M}$ be an o-minimal structure over the reals. A hybrid system $H = (S, S_0, S_F, F, E, I, G, R)$ is *$\mathcal{M}$-definable* if $Q \subseteq \mathbb{R}$ is a definable set[1] and the sets $S_0, S_F$, the mappings $I, G, R$, and the relation $T := \{(q, \vec{x}, \vec{y}) \mid (q, \vec{x}) \xrightarrow{c} (q, \vec{y})\}$ are definable in $\mathcal{M}$. A hybrid system is *o-minimal* if it is definable in some o-minimal structure over $\mathbb{R}$. An interpreted hybrid system $(H, \Pi)$ is *$\mathcal{M}$-definable* if $H$ is $\mathcal{M}$-definable, and for every $\pi$ in the signature, the set $\Pi^{-1}(\pi)$ is definable.

**Theorem 12** *Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ be such that its expansion with $+, \cdot, 0, 1$ is a decidable o-minimal structure. Then the restriction of the LTL-model-checking problem for hybrid systems to $\mathcal{M}$-definable interpreted hybrid systems is decidable.*

*Furthermore, if $\mathcal{M} = \mathbf{R}_{\text{lin}}$ or $\mathbf{R}$, for every fixed LTL-formula $\varphi$ and $n \geq 1$, the restriction of the LTL-model-checking problem problem to $\mathcal{M}$-definable interpreted hybrid systems of dimension $n$ can be solved in* PTIME.

*Proof sketch.* We code hybrid systems as definable databases, and show that for every LTL-formula $\varphi$, there exists and can be effectively found an $L_{\text{PATH}}$ formula $\varphi^*$ such that, for every interpreted hybrid system $(H, \Pi)$, we have $(H, \Pi) \models \varphi^*$ iff for every trajectory $\tau$ of $H$, $\Pi(\tau)$ satisfies $\varphi$. Then the result follows from Theorem 10. □

## 5 Conclusion

Reachability between points in a region is a fundamental notion in spatial reasoning. A tool for analyzing spatial information – a spatial database, or a verification tool for systems that manipulate spatial data – has to somehow take reachability and connectivity into account. From previous work it appeared that incorporating reachability into a spatial language might be fundamentally incompatible with the use of constraint-based representations. Our first results here showed that this is not the case. Instead of attempting to approach connectivity through the use of some discrete recursion mechanism, we added reachability and other topological operators *directly*, and showed that this leads to closed languages. We then tackled the question of getting tractable, closed languages that can express the reachability queries of interest. The language $L_{\text{PATH}}$ has a lot of what one wants in a spatial query language. In addition to the positive results on the data complexity, expressiveness, and closure, we think $L_{\text{PATH}}$ is interesting as a synthesis of the temporal languages for verification of discrete systems with first-order constraint query languages.

Although we approached $L_{\text{PATH}}$ from the point of view of spatial databases, it could also be seen as a general language for stating path properties of systems that are defined from semi-algebraic or semi-linear objects. Because of this, it is possible to compare it with languages for specifying properties of real-time or hybrid systems. We gave an example of how to model one specification formalism for hybrid systems within $L_{\text{PATH}}$.

We do not claim that $L_{\text{PATH}}$ is a practical query language for connectivity queries in spatial databases – it still remains to find a more natural syntax, and to get query evaluation algorithms that run tractably on real applications. In addition, we do not have many results on the expressivity of $L_{\text{PATH}}$ (beyond the PTIME complexity bound), and we know even less about languages $\text{FO} + \mathbb{T}$. We conjecture that $\text{FO} + \mathsf{C} \subset \text{FO} + \mathsf{Conn} \subset L_{\text{PATH}}$, but we know of no techniques for proving separation results of this kind.

*Remark* C. Giannella and D. Van Gucht independently discovered the closure of FO+LIN and FO+POLY under connectivity operators $\mathsf{C}$ and $\mathsf{Conn}$ [13] (our Corollary 4, Proposition 6, and a remark preceding that Proposition). Their proof uses cylindrical algebraic decomposition instead of Local Triviality, and consequently cannot show the closure under all topological properties as we do here; however, for the particular case of connectivity operators they lower the complexity bounds from PTIME to NC.

**Acknowledgement** We thank Saugata Basu, Mihalis Yannakakis, and Thomas Wilke for helpful discussions.

## References

[1] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] F. Afrati, S. Cosmadakis, S. Grumbach and G. Kuper. Linear vs. polynomial constraints in database query languages. In *PPCP'94*, pages 181–192.

[3] R. Benedetti and J.-J. Risler. *Real Algebraic and Semi-algebraic Sets.* Hermann, Paris, 1990.

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

[1] We assume that there exists a finite set of $card(Q)$ definable constants. This is certainly true for all structures of practical interest like $\mathbf{R}_{\text{lin}}$ and $\mathbf{R}$; otherwise one can restate the definition by talking about definability of restrictions of sets like $S, S_F, T$, etc for each $q \in Q$.

[5] M. Benedikt, M. Grohe, L. Libkin and L. Segoufin. Reachability and connectivity queries in constraint databases. Bell Labs Technical Memo, 1999.

[6] M. Benedikt and L. Libkin. Relational queries over interpreted structures. *J. ACM*, to appear. Extended abstract in *PODS'97*, pages 87–98.

[7] M. Ben-Or, D. Kozen and J. Reif. The complexity of elementary algebra and geometry. *JCSS* 32 (1986), 251–264.

[8] J. Bochnak, M. Coste, M.-F. Roy. *Real Algebraic Geometry.* Springer Verlag, 1998.

[9] B.F. Caviness and J.R. Johnson, Eds. *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Springer Verlag, 1998.

[10] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Autom. Theor. Form. Lang*, Springer LNCS 33, Springer Verlag, Berlin, 1975, 134–183.

[11] E. A. Emerson. Temporal and modal logic. Chapter 16 of Volume B of *Handbook of Theoretical Computer Science*, Elsevier, 1990.

[12] F. Geerts and B. Kuijpers. Expressing topological connectivity of spatial databases. In *DBPL'99*.

[13] C. Giannella and D. Van Gucht. Adding a path connectedness operator to FO + poly (linear). Technical Report, Indiana Univ., Nov. 1999.

[14] E. Grädel, S. Kreutzer. Descriptive complexity theory for constraint databases. In *CSL'99*, pages 67–81.

[15] S. Grumbach, G. Kuper. Tractable recursion over geometric data. In *CP'97*, pages 450–462.

[16] J. Heintz, M.-F. Roy and P. Solernó. Description of connected components of a semialgebraic set in single exponential time. *Discrete and Computational Geometry*, 11 (1994), 121–140.

[17] T. A. Henzinger. The theory of hybrid automata. In *LICS'96*, pages 278–292.

[18] N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.* 16 (1987), 760–778.

[19] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *JCSS*, 51 (1995), 26–52.

[20] B. Kuijpers, J. Paredaens and J. Van den Bussche. On topological elementary equivalence of spatial databases. In *ICDT'97*, pages 432–446.

[21] B. Kuijpers and M. Smits. On expressing topological connectivity in spatial datalog. In *CDB'97*, pages 116–133.

[22] B. Kuijpers and J. Van den Bussche. On capturing first-order topological properties of planar spatial databases. In *ICDT'99*, pages 187–198.

[23] G. Kuper, L. Libkin and J. Paredaens, eds. *Constraint Databases.* Springer Verlag, 2000.

[24] G. Lafferriere, G. Pappas and S. Sastry. A new class of decidable hybrid systems. In *Hybrid Systems 1999*, LNCS 1569, Springer, 1999, pages 137–151.

[25] G. Lafferriere, G. Pappas and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, to appear.

[26] C. Papadimitriou, D. Suciu and V. Vianu. Topological queries in spatial databases. In *PODS'96*, pages 81–92.

[27] J. Paredaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over the reals. *SIAM J. Comput.* 27 (1998), 1747–1763.

[28] A. Pillay, C. Steinhorn. Definable sets in ordered structures. III. *Trans. AMS* 309 (1988), 469–476.

[29] P. Revesz. Datalog and constraints. In [23], pages 155–170.

[30] L. Segoufin and V. Vianu. Querying spatial databases via topological invariants. In *PODS'98*, pages 89–98.

[31] L. van den Dries. *Tame Topology and O-minimal Structures.* Cambridge, 1998.

[32] A.J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function. *J. Amer. Math. Soc.* 9 (1996), 1051–1094.