Querying Spatial Databases via Topological Invariants

Luc Segoufin*
I.N.R.I.A.
B.P. 105
78153 Le Chesnay CEDEX
France
Luc.Segoufin@inria.fr

Victor Vianu*
CSE 0114
U.C. San Diego
La Jolla, CA 92093-0114
U.S.A.
vianu@cs.ucsd.edu

Abstract

The paper investigates the use of topological annotations (called topological invariants) to answer topological queries in spatial databases. The focus is on the translation of topological queries against the spatial database into queries against the topological invariant. The languages considered are first-order on the spatial database side, and *fixpoint* and first-order on the topological invariant side. In particular, it is shown that *fixpoint* expresses precisely the PTIME queries on topological invariants.

1 Introduction

Spatial data is an increasingly important part of database systems. It is present in a wide range of applications: geographic information systems, video databases, medical imaging, CAD-CAM, VLSI, robotics, etc. Different applications pose different requirements on query languages and therefore on the kind of spatial information that is needed. For example, in some cases the precise distance between points is important, while in other applications only *topological* relationships is of interest. Such differences in scope and emphasis are crucial, as they affect the data model, the query language, and performance. In this paper we focus on the representation and querying of topological properties of spatial databases.

Motivated primarily by geographic information systems, we use a spatial model that speaks about *regions* in the two-dimensional plane. Regions are specified by inequalities involving polynomials with rational coefficients, as done in constraint databases (such regions are called *semi-algebraic*). Topological properties of regions are those that are invariant under homeomorphisms of the plane. This means, intuitively, that continuous deformations and reflections of the spatial instance do not affect satisfaction of the property. For example, the property "the intersection of regions P and Q is connected" is a topological property. On the other hand, the property "P lies above Q" is *not* topological. In previous work [PSV96] it was shown that topological properties of semi-algebraic regions in a spatial database can be completely summarized by an annotation presented in classical relational database form, called the *topological invariant* of the database[1]. Moreover, the topological invariant of a semi-algebraic database can be constructed very efficiently – in NC.

Suppose that a topological query is posed against the spatial database. In principle, the query can be answered by another query posed against the topological invariant. Since the topological invariant is in most cases much simpler than the full spatial data, this strategy is likely to be more efficient. In order for this to work, topological queries in the spatial query language need to be *effectively translated* into queries in some query language for topological invariants. There are two components to this question:

- What language $\mathcal{L}_{inv}$ on topological invariants is needed in order to answer the topological queries formulated in a given query language $\mathcal{L}_{spatial}$ on spatial databases?
- Is there an effective, uniform translation of topological queries in $\mathcal{L}_{spatial}$ into queries in $\mathcal{L}_{inv}$? What is the complexity of the translation?

To answer the first question, it is useful to understand the expressiveness of various query languages on topological invariants. Topological invariants have special structure, so with some luck they might be better behaved than arbitrary relational databases. This is fully confirmed by our first result: we show that *fixpoint* expresses precisely the PTIME queries over topological invariants. This should be contrasted with the situation on arbitrary structures, where *fixpoint* falls short of capturing PTIME (indeed, it is conjectured that there is *no language* capturing PTIME). This result is very helpful in answering the first question above, since it makes *fixpoint* a natural target for the translation of topological queries on spatial databases. In the broader context of the theory of query languages, the result is significant because it identifies topological invariants as a class of finite structures of practical interest which is very well-behaved with respect to descriptive complexity.

*This work was partly performed while the first author was visiting U.C. San Diego and partly while the second author was visiting I.N.R.I.A. V.Vianu was supported in part by the NSF under grant number IRI-9221268.

[1] A similar invariant for isotopy-invariant properties is presented in [KPV95], see related work.

With respect to the second question, we focus on the translation problem for first-order queries. A common language for constraint spatial databases is $FO(\mathbb{R}, <)$. We consider primarily this language on the spatial database side. On the topological invariant side, *fixpoint* is a possible translation target, as suggested by the earlier expressiveness result. Indeed, we show that topological $FO(\mathbb{R}, <)$ queries can be uniformly translated in linear time into *fixpoint* queries on the invariant. However, another natural candidate target for the translation is FO. The translation problem now becomes much harder, and we solve it in the special case of single-region databases. However, the region can be highly complex, so the result is fairly general and provides considerable insight into the technical issues involved in the translation.

Interestingly, even when both FO and *fixpoint* can be used as targets of the translation of topological $FO(\mathbb{R}, <)$ queries, there is a significant difference in the complexity of the translation: the complexity is 3-EXPTIME[2] in the quantifier depth of the input query when the target is FO, but it goes down to linear time in the size of the query when the target is *fixpoint*. This suggests an interesting trade-off between the expressive power of the target query language over the topological invariant and the complexity of the translation.

Despite the high complexity of the translation, we argue that translation wins over direct evaluation in many practical situations, due to the much smaller size of the invariant relative to the full instance. This is backed up by several sets of cartographic data from Sequoia 2000 [Seq] and the French National Geographical Institute [IGN].

Related work. Work in spatial databases has focused on developing models and query languages targeted to various application domains, as well as appropriate data structures and efficient evaluation techniques. We refer to [Par95] for a survey of the field emphasizing geographic information systems.

Our topological invariants contain information similar to the PLA model proposed by the U.S. Census Bureau, which contains topological properties on points, lines, and areas [Cor79, Par95]. The invariants can be viewed as an augmentation of the PLA model. As mentioned earlier, the invariants proposed in [KPV95] come closest to the topological invariants we consider, but capture isotopy-generic information. Query languages are not considered in [KPV95]. Note that partial topological annotations have been used for some time in the GIS community to speed up query evaluation (e.g. in the ARC/INFO system [M85, M89]).

Various notions of G-invariance (or G-genericity) for different groups G of permutations are discussed in [Par+94]. They propose a spatial database model that includes spatial and thematic information, and propose a calculus and an equivalent algebra.

Much of the formal work related to spatial databases focuses on *constraint databases*, consisting of relations whose tuples represent semi-algebraic regions, specified by polynomial inequalities. Such databases and corresponding query languages were first considered in [KKR90]. In particular, they investigate the question of when the answer of a query on a constraint database is representable as a constraint database. Their results are based on quantifier elimination in the first-order theory of the reals [Tar51]. Further work on expressiveness of query languages on constraint databases includes [GS97, GS98, GST94, Par+95, BDLW96].

The paper is organized as follows. The spatial model used in the paper, as well as topological invariants, several relational and spatial query languages, and Ehrenfeucht-Fraissé games, are reviewed in the Preliminaries. Section 3 contains the result on the capture of PTIME by *fixpoint* on topological invariants, and related results. Section 4 presents the results on the translation of spatial topological $FO(\mathbb{R}, <)$ queries into *fixpoint* and FO queries on topological invariants.

2 Preliminaries

Practical spatial databases (such as geographic systems) mix thematic and spatial information. Answers to queries can also be multi-sorted. Since our focus is on the spatial aspect, we adopt a simplified model where the only thematic information consists of region names. Also, for the sake of simplicity and uniformity, we only consider boolean queries, defining *properties* of sets of regions. We consider only regions in the two-dimensional space.

We use the following model for spatial databases. We assume given an infinite set **names** (consisting of names of regions). A spatial database *schema* is a finite subset of **names**. An *instance* I over a schema Reg is a mapping from Reg to subsets of $\mathbb{R}^2$. For each $r \in Reg$, $I(r)$ provides a set of points called the *extent of* r. We generally refer to a set of points in the plane as a *region*. In practice, each $I(r)$ is finitely specified, although this may be transparent to the user. In this paper, all regions considered are compact, connected (i.e. they have connected boundary), and specified by a disjunction of conjunctions of polynomial inequalities with rational coefficients (such regions are usually referred to as *semi-algebraic*). In the following, the term *region* will be used in the restricted manner just described, unless otherwise specified.

As discussed earlier, we are interested here in topological properties of spatial instances. Two instances I, J over a spatial schema Reg are *topologically equivalent* iff there exists a bijection $\lambda : \mathbb{R}^2 \to \mathbb{R}^2$ such that both λ and λ^{-1} are continuous and for each $r \in Reg$, $\lambda(I(r)) = J(r)$. A property of spatial instances is *topological* if it is invariant under homeomorphisms, that is for all topologically equivalent instances I, J over a given schema, I has the property iff J has the property.

Topological invariants In [PSV96] it was shown that one can efficiently compute from a given semi-algebraic spatial instance I a finite relational structure $top(I)$ called the *topological invariant of* I, that describes completely the topological properties of I. The spatial model used in [PSV96] is slightly different from the one we use: regions are assumed to be homeomorphic to the unit disk, so are open and have dimension two. In contrast, our regions are closed and may have dimension zero, one, or two (the change is motivated by geographic information systems, where such regions are common). However, the construction and results of [PSV96] pertaining to the topological invariant are easily adapted to the model used here. We briefly describe the construction

[2] K-EXPTIME denotes the functions computable in time bounded by a tower of k exponentials raised to a polynomial in the size of the input, and similarly for space. Thus, for 3-EXPTIME the bound is $2^{2^{2^{n^h}}}$ for some h, where n is the size of the input.

of the invariant and the results.

The invariant is constructed from a *maximum topological cell decomposition* of the spatial instance. A *topological cell decomposition* of I is a partition of $\mathbb{R}^2$ into finitely many subsets called *cells*, such that for every homeomorphism λ of $\mathbb{R}^2$, if $\lambda(I) = I$ then $\lambda(c)$ is a cell for every cell c. It is easily verified that for each spatial instance I there exists a unique maximal (in terms of number of cells) topological cell decomposition. The maximum topological cell decomposition can be constructed from a semi-algebraic spatial instance in NC, using results on *cell complexes* obtained in [KY85]. We summarize their approach, slightly adapted to our context. Given a semi-algebraic spatial instance I over a schema *Reg*, a *sign assignment* is a mapping $\sigma : Reg \rightarrow \{o, -, \partial\}$, and the *sign class* of σ is the set $I^\sigma \stackrel{\text{def}}{=} \bigcap_{r \in Reg} r^{\sigma(r)}$, where r^o is the interior of r, r^∂ is the boundary, and r^- is the exterior. A *cell complex* for I is a partition of $\mathbb{R}^2$ into finitely many, non-empty, pairwise disjoint regions, called *cells*, such that:

1. each cell is homeomorphic to $\mathbb{R}^0, \mathbb{R}^1$ or $\mathbb{R}^2 -$ {a finite set of points}. The *dimension* of a cell is defined in the obvious manner.
2. the closure of each cell is a union of other cells;
3. each cell is included in some sign class I^σ.

It is shown in [KY85] that a cell complex can be constructed from a given semi-algebraic spatial instance in NC. One can further show that the maximum topological cell decomposition can be constructed from the cell complex obtained in [KY85], and the overall complexity remains NC.

The topological invariant for a spatial instance I is build up from the maximum topological cell decomposition for I. Cells of dimension 0,1 and 2 are called vertices, edges, and faces, respectively. The topological invariant associated to spatial instances over a schema *Reg* is a finite structure consisting of the following relations (their meaning is explained intuitively):

1. unary relations *Vertex*, *Edge*, *Face*, and *Exterior -face* providing the cells of dimension 0, 1, 2, and a distinguished face of dimension 2 called the *exterior face*.
2. *Endpoints* is a ternary relation providing endpoint(s) for edges.
3. *Face-Edge* is a binary relation providing, for each face (including the exterior cell), the edges on its boundary.
4. *Face-Vertex* is a binary relation providing, for each face (including the exterior cell), the vertices adjacent to it.
5. for each region name $p \in Reg$, a unary relation p providing the set of cells contained in region p.
6. *Orientation* is a 4-ary relation providing the clockwise and counterclockwise orientation of edges incident to each vertex. More precisely, $(\leftarrow, v, e_1, e_2) \in$ *Orientation* iff v is a vertex and e_1, e_2 are clockwise consecutive edges incident to v, and $(\hookrightarrow, v, e_1, e_2) \in$ *Orientation* iff v is a vertex and e_1, e_2 are counterclockwise consecutive edges incident to v.

Let *inv*(*Reg*) denote the above schema and let *top* denote the mapping associating to each spatial instance I over *Reg* its topological invariant over *inv*(*Reg*).

The main result on the topological invariant is the following.

Theorem 2.1 *Let* Reg *be a spatial database schema.*
(i) The mapping top *associating to each spatial database instance over* Reg *its topological invariant, is computable in polynomial time (and* NC*) with respect to the size of the representation of I.*
(ii) For all spatial instances I, J over Reg, I and J are topologically equivalent iff top(I) and top(J) are isomorphic.

Also useful is the following result, which says that the invariant can be efficiently inverted. A *linear spatial instance* is a semi-algebraic instance defined by linear inequalities.

Theorem 2.2 *For each spatial instance I there exists a topologically equivalent linear spatial instance J computable in polynomial time from top(I).*

Some query languages We assume familiarity with classical relational query languages such as FO (relational calculus), Datalog, Datalog$^\neg$, and the *fixpoint* and *while* queries (see [AHV95]). Recall that the *fixpoint* queries are expressed by various languages such as inflationary Datalog$^\neg$ and inflationary fixpoint logic FO+μ^+ [AHV95]. It is well known that *fixpoint* expresses precisely PTIME [I86, V82] and *while* expresses exactly PSPACE [V82] *on ordered databases*. Without order, this is not the case: for example, neither *fixpoint* nor *while* can express the *parity* query on unary relations (this asks if the number of elements in the relation is even or odd).

Most of the languages previously proposed for spatial databases, including constraint query languages [KKR90, GS98, GST94, Par+95, BDLW96], have first-order syntax and use variables ranging over *numbers* (reals or rationals), or over *points*. We adapt the classical definitions of these languages to our framework. Let *Reg* be a spatial database schema. By slight abuse of notation we denote also by *Reg* the first-order signature consisting of a binary relation for each region name in *Reg*. The language $FO(\mathbb{R}, <)$ is first-order logic using region names (viewed as binary relations), variables ranging over $\mathbb{R}$, and the binary relation $<$ interpreted as the usual order on $\mathbb{R}$. It was shown in [KKR90] that every $FO(\mathbb{R}, <)$ query can be evaluated in NC (relative to the size of the representation of a given semi-algebraic spatial instance). A useful variation is first-order logic where variables range over points in $\mathbb{R}^2$, and where the order $<$ is replaced by two order relations $<_x$ and $<_y$ with the following meaning: $p <_x q$ iff the x-coordinate of p $<$ the x-coordinate of q, and similarly for $<_y$. Note also that the schema consists of region names viewed as unary relations rather than binary. The point-based language just described is denoted by $FO(\mathbb{P}, <_x, <_y)$. Its complexity is also NC.

What is the connection between $FO(\mathbb{R}, <)$ and $FO(\mathbb{P}, <_x, <_y)$? Clearly, $FO(\mathbb{R}, <)$ subsumes $FO(\mathbb{P}, <_x, <_y)$, and can express queries that $FO(\mathbb{P}, <_x, <_y)$ cannot, such as: "does region P intersect the diagonal ?" as $\exists x\ P(x, x)$. However, it is shown in [PSV96] (and the result easily carries over to our spatial model) that the two languages express *precisely the same topological properties* of spatial instances. The fragments of the two languages expressing topological properties

are denoted by $FO_{top}(\mathbb{R}, <)$ and $FO_{top}(\mathbb{P}, <_x, <_y)$, respectively. It is easy to see that these fragments are not effective: it is undecidable whether a sentence in $FO(\mathbb{R}, <)$ or in $FO(\mathbb{P}, <_x, <_y)$ is topological. In this paper we assume it given that queries arising in certain applications are topological, and we do not deal with the issue of verifying or enforcing this property. To our knowledge, it is open whether there exist effective syntactic subsets of $FO(\mathbb{R}, <)$ and $FO(\mathbb{P}, <_x, <_y)$ that express precisely their topological fragments.

Ehrenfeucht-Fraissé games Consider first-order logic FO over a given vocabulary. Let FO^r denote the FO sentences of quantifier depth r. Recall that the quantifier depth $qd(\varphi)$ is defined inductively by $qd(\varphi) = 0$ if φ is quantifier-free, $qd(\varphi \vee \psi) = qd(\varphi \wedge \psi) = max\{qd(\varphi), qd(\psi)\}$, $qd(\neg\varphi) = qd(\varphi)$ and

$$qd(\exists x\varphi) = qd(\forall x\varphi) = qd(\varphi) + 1.$$

Two structures are FO^r-*equivalent* if they satisfy precisely the same FO^r sentences. There is a very useful characterization of FO^r-equivalence in terms of the *Ehrenfeucht-Fraissé game* on structures. The game of length r is played by two players, Spoiler and Duplicator. A round in the game has r moves. In each move, Spoiler picks an element in one of the structures and Duplicator responds by picking an element in the opposite structure. Duplicator wins the round if structures restricted to the chosen elements are isomorphic. We say that Duplicator *has a winning strategy for the game of length* r if he can win every round of the game no matter how Spoiler plays. The main result on the Ehrenfeucht-Fraissé game of length r (henceforth called FO^r-game) is that two structures are FO^r-equivalent iff Duplicator has a winning strategy for the FO^r-game.

As an example, consider the language $FO(\mathbb{P}, <_x, <_y)$ over spatial database schema *Reg*. Suppose I, J are spatial instances over *Reg* and consider a round of the $FO^r(\mathbb{P}, <_x, <_y$)-game where points $p_1, \ldots, p_r$ are picked in I and $q_1, \ldots, q_r$ are picked in J. Duplicator wins this round iff $p_i \in P \Leftrightarrow q_i \in P$ for every $P \in Reg$ and p_i, p_j are in the same order relative to $<_x$ and $<_y$ as q_i, q_j, for $1 \leq i, j \leq r$. By the above characterization, Duplicator has a winning strategy for the $FO^r(\mathbb{P}, <_x, <_y)$-game on I, J iff I and J are equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$.

The existence of a winning strategy for Duplicator is in many cases hard to prove. Of great help are results providing sufficient conditions. We will use one such condition, provided in [FSV95]. Given a graph G, the d-neighborhood of a vertex a is defined as the subgraph of G whose vertices are those at distance at most d from a. Two graph structures I and J are called (d, m)-equivalent if for each d-neighborhood type τ either I and J have exactly the same number of vertices of d-neighborhood type τ, or both graphs have at least m vertices of type τ. An (d, m)-equivalence class is described by its (d, m)-*type*, specifying for each isomorphism type of a d-neighborhood the number of times it occurs, truncated to m if the number of occurrences is larger. The following useful result is shown in [FSV95].

Theorem 2.3 *If two graphs I and J are $(3^r, r2^{3^r-1})$-equivalent then Duplicator has a winning strategy for the FO^r-game on I and J, and therefore I and J are equivalent with respect to FO^r.*

We will be interested in applying the above theorem to special graphs which are unions of colored cycles. Due to the particular shape of neighborhoods, the theorem can be tightened as follows.

Corollary 2.4 *If two graphs I and J consisting of unions of colored cycles are $(3^r, r3^r)$-equivalent then they are equivalent with respect to FO^r.*

3 Fixpoint queries on topological invariants

As discussed earlier, the topological invariant of a spatial instance can be viewed as an annotation summarizing precisely the topological properties of the spatial instance. Thus, all topological queries can be answered by queries posed against the invariant rather than on the raw spatial data. This can be much more efficient, since the invariant is in most cases much smaller than the original spatial instance. The invariant is a classical relational database, so it can be queried by classical relational queries. However, invariants are not arbitrary databases – they have a special structure, which may engender special properties. In this section we show that topological invariants are especially well behaved with respect to descriptive complexity: the *fixpoint* queries capture precisely PTIME over this class of structures. This positive result should be contrasted with the situation on arbitrary structures, where there is a large gap between *fixpoint* and PTIME (in the absence of order). Moreover, it is conjectured that there is no language capturing PTIME on arbitrary structures.

The proof that *fixpoint* captures PTIME on topological invariants is conceptually easy, but requires some careful development. We use the classical result that *fixpoint* captures PTIME on ordered structures [I86, V82]. Thus, for each PTIME query over topological invariants there exists a *fixpoint* query $\varphi(\leq)$ which expresses the PTIME query given a total order $\leq$ on the universe of the invariant. The problem of course is that topological invariants are not ordered structures – in fact they are not even rigid – so an order $\leq$ is not directly available. The key to the proof is to observe that there is a standard way to traverse each connected component of the topological invariant, once a *constant* number of vertices and/or edges have been fixed, together with an orientation ($\hookleftarrow$ or $\hookrightarrow$). This allows to construct by a *fixpoint* query a *polynomial number of orderings*, each of which is identified by a tuple of constant arity. Since the number of connected components is bounded by the number of regions in the schema, we can put together the orders for each connected component and obtain a polynomial number of orders of the entire invariant. Lastly, $\varphi(\leq)$ is "run simultaneously" on all of the orderings. Since $\varphi(\leq)$ is order independent, all of the "runs" produce the answer to the query.

We now describe the construction in more detail. We break down the construction in several sequential stages, and for each we outline the construction of a *fixpoint* query. Since *fixpoint* is closed under composition, we can then put together the queries for each stage and obtain a final *fixpoint* query that computes the given PTIME query on the invariant. Consider a PTIME query over the invariant and let $\varphi(\leq)$ be a *fixpoint* query that computes it using a total order $\leq$ on the universe (vertices, edges, faces, $\hookleftarrow$ and $\hookrightarrow$) of the invariant.

The *skeleton* of a topological invariant is the graph whose vertices and edges are those of the invariant, and an edge

connects two vertices iff the vertices are its endpoints according to relation *Endpoints* in the invariant. A *connected component* of the topological invariant is a connected component of the skeleton of the invariant. Recall that all regions are connected, so each connected component of the invariant consists of the boundaries of a set of regions. Thus, there is a partition π of the set of region names into equivalence classes defined by membership of the boundaries into the same connected component. Clearly, for each partition π there exists an FO query σ_π which verifies that π is the partition of region names corresponding to the given topological invariant.

We will also need a formula defining the set of vertices, edges, and faces of each connected component. For vertices and edges this can be easily done by an FO formula. More care is needed for faces, since the boundary of a face may intersect several connected components (this happens when a connected component is embedded in a face of another connected component). To associate unambiguously a face to a connected component, one can first define a partial order on connected components based on their distance from the external face (i.e. the shortest sequence $f_0, e_0, f_1, e_1, \ldots f_k, e_k$ such that f_0 is the exterior face, e_k is in the given connected component, and e_i is adjacent to $f_i, 0 \leq i \leq k$). It is easily seen that for each face other than the exterior face f_0 there exists a unique connected component at minimal distance from f_0 intersecting the boundary of the face. We then say that the face belongs to that connected component, and this can be stated by a *fixpoint* formula. In summary, one can construct a *fixpoint* query $comp_i(x)$ saying that x is a vertex, edge, or face in connected component i.

Consider a connected component of the topological invariant. An edge is called a *proper edge* if it connects distinct vertices. We will show that one can construct by a *fixpoint* query a total order on the vertices, edges, and faces of the connected component, once a vertex, an adjacent proper edge, and an orientation $\omega \in \{\hookleftarrow, \hookrightarrow\}$ have been fixed. This applies to connected components which have at least one proper edge (and therefore each vertex is connected by some proper edge to another vertex). The special case where there is no proper edge requires special treatment, but presents no difficulty. We can show the following.

Lemma 3.1 *There exists an $FO + \mu^+$ formula $\alpha(\omega, v, e, x, y)$ such that for each orientation ω, vertex v and proper edge e adjacent to v, the set*

$$\leq_{\omega,v,e}(x, y) = \{\langle x, y\rangle \mid \alpha(\omega, v, e, x, y)\}$$

is a total order on the vertices, edges, and faces of the connected component of v.

Once the query α is constructed, we can use it to generate a set of total orders on the entire invariant as follows. Basically, we need to put together the orders for the connected components of the invariant in some arbitrary way determined by some fixed order of the region names in the schema, which induces an order on the connected components. Suppose π is the partition of region names determined by the connected components of the invariant. The set of total orders on the invariant corresponding to partition π is then obtained by placing in increasing order: (i) the elements denoting the orientations, (ii) the external face, and (iii) the ordered vertices, edges, and faces of each connected component, in increasing order of the components. This is accomplished by the FO+μ^+ formula:

$$\begin{aligned}(\dagger)\quad & \leq_\pi(\omega, v_1, e_1, \ldots, v_k, e_k, x, y) = \\ & \mathit{Is\text{-}Orientation}(\omega) \\ & \wedge \textstyle\bigwedge_{1\leq i\leq k} \mathit{vertex\text{-}edge}_i(v_i, e_i) \\ & \wedge[(x = \omega) \\ & \vee(\mathit{Is\text{-}Orientation}(x) \wedge y \neq \omega) \\ & \vee(\mathit{Exterior\text{-}face}(x) \wedge \neg \mathit{Is\text{-}Orientation}(y) \\ & \textstyle\bigvee_{1\leq i\leq k} \alpha(\omega, v_i, e_i, x, y) \\ & \vee \textstyle\bigvee_{1\leq i<j\leq k}(comp_i(x) \wedge comp_j(y))]\end{aligned}$$

where

$$\mathit{Is\text{-}Orientation}(x) = \exists v \exists e_1 \exists e_2 \mathit{Orientation}(x, v, e_1, e_2).$$

Thus, $\leq_\pi$ has arity $2k + 3$ where k is the size of partition π. The last two columns in the result provide a total order on the invariant, for each fixed tuple over the first $2k + 1$ columns. Note that k is bounded by the size of *Reg* so is a constant.

Once $\leq_\pi$ is constructed, the *fixpoint* query $\varphi(\leq)$ can be "run" in parallel on the orders provided by $\leq_\pi$. Because the query $\varphi(\leq)$ is order invariant, the answer is the same for all orderings. This yields the main result:

Theorem 3.2 Fixpoint *expresses exactly the* PTIME *queries on topological invariants.*

The above result can be extended to languages subsuming *fixpoint*, such as *while*. A technique similar to the above allows to show:

Corollary 3.3 While *expresses exactly the* PSPACE *queries on topological invariants.*

At this point, it may be tempting to believe that descriptive complexity results on ordered structures transfer uniformly to topological invariants. However, this is far from obvious. For example, it is known that semi-positive Datalog, Datalog$^\neg$ with stratified semantics, and inflationary Datalog$^\neg$ are equivalent on ordered structures and express PTIME (see [AHV95]). However, it remains open if this holds on topological invariants. Indeed, the *fixpoint* query constructing the orderings on topological invariants require careful bookkeeping which, on the face of it, requires the full power of inflationary Datalog$^\neg$ or equivalently FO+μ^+. In particular, negation is applied recursively in our construction of the orderings. It remains open whether the construction can be achieved in a more restrictive language such as stratifiable or semi-positive Datalog$^\neg$.

Remark Theorem 3.2 relies crucially on two features of our spatial model: (i) each region is connected, and (ii) region names are part of the database schema, and so are fixed. Suppose first that (i) is relaxed so that regions are not required to be connected. In this case there is an unbounded number of connected components, and *fixpoint* cannot express PTIME queries such as: *Is there an even number of connected components?* However, this difficulty can be overcome by adding counting to *fixpoint*. Indeed, we can show that in this case *fixpoint + counting* (as defined in [GO93] and implicit in [I87]) expresses precisely the PTIME queries

on topological invariants [Seg]. The idea of the proof is to construct using *fixpoint + counting* an isomorphic copy of the invariant over the auxiliary ordered domain provided by *fixpoint + counting*, then use the fact that *fixpoint* expresses PTIME on ordered databases [I86, V82].

Now suppose that (ii) is relaxed so that region names are part of the instance rather than fixed in the schema. Then *fixpoint + counting* fails to capture PTIME. This is shown by representing arbitrary graphs as spatial instances where each edge is identified by a region name, and using the fact that *fixpoint + counting* fails to capture PTIME on arbitrary graphs [CFI92].

PTIME topological queries on spatial instances The capture of PTIME by *fixpoint* on topological invariants has consequences for the capture of the PTIME *topological queries on spatial instances*. Indeed, Theorem 3.2 together with Theorem 2.2 imply that there exists a language expressing precisely the PTIME topological queries on spatial instances. Queries in the language consist simply of a *fixpoint* query applied to the topological invariant of the given spatial instance. Clearly, all such queries are topological and are computable in PTIME in the size of the representation of the spatial instance. To see that the converse is true, let φ be a topological PTIME query on spatial instances. Let $inv(\varphi)$ be the query on topological invariants which does the following:

1. on input T, compute from T a linear spatial instance J such that $top(J) = T$. This can be done in PTIME by Theorem 2.2.
2. compute $\varphi(J)$.

Clearly, the query $inv(\varphi)$ is in PTIME on topological invariants, so it is expressed by some *fixpoint* query ψ. Since φ is topological and I and J are topologically equivalent, we have that $\varphi(I) = \varphi(J) = \psi(top(I))$ for every spatial instance I. Thus, φ is expressed by a query in the language.

It is worth noting that the capture of topological PTIME on spatial instances is not simply a direct consequence of Theorems 2.1 and 3.2, as one might first be tempted to believe. Indeed, this relies crucially on the fact that topological invariants can be efficiently inverted (Theorem 2.2). To understand the importance of this fact, it is useful to make a parallel with invariants for finite structures undistinguishable by first-order logic sentences with k variables, called k-invariants. Like topological invariants, k-invariants can be computed efficiently by a *fixpoint* query (see [AV95, DLW95]). Furthermore, the k-invariants are ordered structures so *fixpoint* captures PTIME on k-invariants. Thus, the analogs of Theorems 2.1 and 3.2 hold. However, this does *not* imply the existence of a logic for PTIME on arbitrary structures. What is missing is the ability to efficiently invert k-invariants [Gro97] – the analog of Theorem 2.2.

Of course, the language just described for topological PTIME is quite artificial, but serves the purpose of showing that there exists a logic for topological PTIME on spatial instances, in the broad sense of Gurevich's definition [Gur84]. It remains open to find a more natural language that captures topological PTIME on spatial instances – perhaps a recursive extension of $FO(\mathbb{R}, <)$. We note that a recursive extension of $FO(\mathbb{R}, <)$ expressing all PTIME queries on spatial instances is presented in [GK97]. However, the language also expresses nontopological queries.

4 Translating spatial queries into queries on the invariant

In this section we study the problem of translating topological queries against spatial databases into queries against the topological invariant. On the spatial database side, we focus on the language $FO(\mathbb{R}, <)$, which is a commonly used language in constraint databases. On the topological invariant side, a natural candidate target for the translation is FO. Most of the results of the section concern the connection between the two languages. Before addressing this problem, we note however that the capture of PTIME by *fixpoint* on topological invariants provides for free an effective translation of topological $FO(\mathbb{R}, <)$ queries into *fixpoint* queries on the invariant. Also, this works in principle for other spatial languages whose topological fragments are in PTIME. Perhaps more interesting is the fact that the translation into *fixpoint* is quite efficient (which turns out not to be the case when the target is FO). We state this result next.

Theorem 4.1 *Let* Reg *be a spatial database schema. There exists a mapping* inv *from* $FO_{top}(\mathbb{R}, <)$ *sentences over* Reg *into* $FO{+}\mu^+$ *sentences over* $inv(Reg)$*, such that:*

1. *for each* $\varphi \in FO_{top}(\mathbb{R}, <)$ *and each spatial instance* I *over* Reg, $\varphi(I) = inv(\varphi)(top(I))$.
2. $inv(\varphi)$ *is computable in linear time in the size of* φ.

Proof Suppose φ is a sentence in $FO_{top}(\mathbb{R}, <)$ over schema *Reg*. The following procedure computes $\varphi(I)$ on input $T = top(I)$:

1. (a) build an instance J over *Reg* such that $top(J) = T$, using the PTIME algorithm provided by Theorem 2.2 for computing J.
   (b) evaluate $\varphi(J)$ using the PTIME algorithm of [KKR90];
2. output the result of evaluating $\varphi(J)$ in 1(b).

This procedure takes time polynomial in the size of the invariant, therefore, from theorem 3.2 it follows that there exists an $FO + \mu^+$ sentence $inv(\varphi)$ implementing the above program, that is $\varphi(I) = inv(\varphi)(top(I))$. Furthermore, the query $inv(\varphi)$ can be effectively constructed from φ. Note that parts 1, 1(a) and (2) of $inv(\varphi)$ are independent of φ so constructing the portions of $inv(\varphi)$ implementing them takes constant time with respect to φ. A careful analysis shows that the portion of $inv(\varphi)$ implementing part 1(b) can be constructed in time linear in the size of φ. □

In the above, as in Theorem 4.3, the translation $inv(\varphi)$ of φ is only guaranteed to be correct if φ is topological. As discussed in Preliminaries, we do not deal here with the issue of verifying or enforcing that φ is indeed topological (which is undecidable). Instead, we assume that this property is guaranteed by the nature of the application, by the query interface, or other extraneous reasons.

In the remainder of the section we consider the connection between $FO_{top}(\mathbb{R}, <)$ and FO_{inv}. We begin with an example that provides some intuition into the difficulties involved in translating $FO_{top}(\mathbb{R}, <)$ sentences into FO_{inv} sentences. Consider the query on schema $Reg = \{P, Q\}$:

"Regions P and Q intersect only on their boundaries."

Clearly, this is a topological property. It can be expressed in $FO_{top}(\mathbb{R}, <)$ by the sentence:

$$(\dagger) \quad \forall x \forall y[(P(x,y) \wedge Q(x,y)) \rightarrow$$

$$(boundary_P(x,y) \wedge boundary_Q(x,y))]$$

where $boundary_P(x,y)$ (and similarly $boundary_Q(x,y)$) is the formula[3]:

$$P(x,y) \wedge \forall x_1 \forall y_1 \forall x_2 \forall y_2[(x_1 < x < x_2 \wedge y_1 < y < y_2) \rightarrow$$

$$\exists x' \exists y'(x_1 < x' < x_2 \wedge y_1 < y' < y_2 \wedge \neg P(x', y')].$$

Clearly, the same property can be expressed by the FO_{inv} sentence over $inv(Reg)$:

$$(\ddagger) \quad \forall u[(P(u) \wedge Q(u)) \rightarrow (Vertex(u) \vee Edge(u))].$$

However, how to get from (†) to (‡) is mysterious. The difficulty is to algorithmically extract the topological meaning of an $FO_{top}(\mathbb{R}, <)$ sentence like (†) that uses nontopological statements involving reals and $<$. We solve the translation problem for the case when Reg contains a single region name; the problem remains open in the general case of several region names. However, recall that our regions can be arbitrarily complicated (as long as they are semi-algebraic, connected and compact), so the result is fairly general and provides considerable insight into the technical issues involved in the translation. In the remainder of the section, spatial database schemas have a single region name unless otherwise specified.

We next describe informally the technical machinery needed to prove the results on the translation of $FO_{top}(\mathbb{R}, <)$ queries into FO_{inv} queries. For technical reasons, it will be useful to first translate $FO_{top}(\mathbb{R}, <)$ queries into the point-based language $FO_{top}(\mathbb{P}, <_x, <_y)$. By [PSV96], this translation can be done in linear time. Thus, it is sufficient to show how to translate $FO_{top}(\mathbb{P}, <_x, <_y)$ queries into FO_{inv} queries. We make use of results in [KPV97] concerning the equivalence of spatial instances (with a single region name) with respect to $FO_{top}(\mathbb{R}, <)$. It is shown there that equivalence of spatial instances with respect to $FO_{top}(\mathbb{R}, <)$ (and therefore $FO_{top}(\mathbb{P}, <_x, <_y)$) is completely characterized by the *cone type* of the instances, that is the multiset consisting of all vertices (cells of dimension zero) and the cyclic list of their adjacent edges and faces (labeled by whether or not they belong to the region). Furthermore, [KPV97] provides a normal form constructing from each instance I an instance $cones(I)$ consisting essentially of the cones of I. For example, given the instance I in Figure 1, the instance $cones(I)$ is represented in Figure 2 (see [KPV97] for the formal definition).

Thus, I and $cones(I)$ satisfy the same $FO_{top}(\mathbb{P}, <_x, <_y)$ sentences and $cones(I)$ is generally much simpler than I. In terms of the invariant, the relevant information about $cones(I)$ can be represented as a set of colored cycles, each of which represents the cyclic list of edges and faces adjacent to a given vertex, according to the orientations. This structure is denoted by $cycles(I)$. For example, $cycles(I)$ for the instance I in Figure 1 is represented in Figure 3. In the figure, a dark node represents a face in the region, and an empty node represents an edge.

It is easy to see that $cycles(I)$ can be computed by first-order means from $top(I)$. Let α_{cycles} be an FO_{inv} formula

[3]With slight abuse of notation.

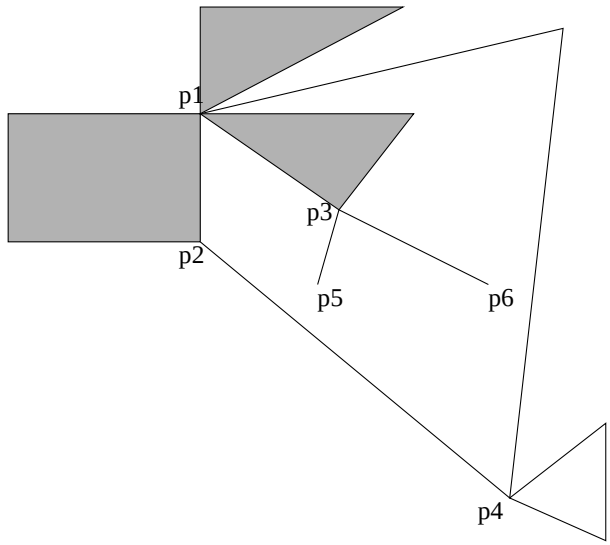

Figure 1: An instance I

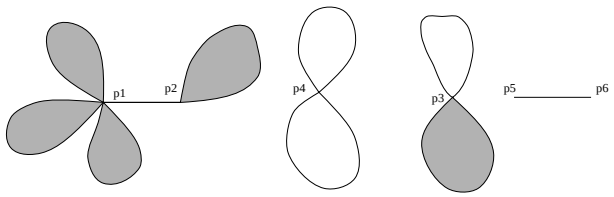

Figure 2: $cones(I)$

defining $cycles(I)$ from $top(I)$. We show the following key fact.

Lemma 4.2 *Let I, J be spatial instances. If $cycles(I)$ and $cycles(J)$ are equivalent with respect to FO^r then I and J are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$.*

Proof (**Sketch**) By [KPV97], I is equivalent to $cones(I)$ with respect to $FO_{top}(\mathbb{P}, <_x, <_y)$, and similarly for J and $cones(J)$. Thus, it is enough to show that $cones(I)$ and $cones(J)$ are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$. We use Ehrenfeucht-Fraissé games. Since $cycles(I)$ and $cycles(J)$ are equivalent with respect to FO^r, Duplicator has a winning strategy for the FO^r-game on these structures. We wish to show that Duplicator has a winning strategy for the $FO^r(\mathbb{P}, <_x, <_y)$-game on $cones(I)$ and $cones(J)$. To facilitate the task, we first apply homeomorphisms that produce representations of $cones(I)$ and $cones(J)$ which are very well-behaved with respect to $<_x$ and $<_y$. Specifically, a homeomorphism can be applied to $cones(I)$ (and similarly to $cones(J)$) so that all vertices are on the x-axis and all

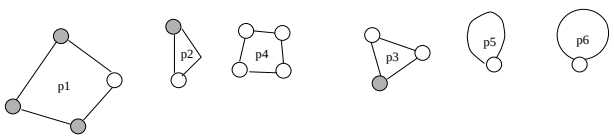

Figure 3: $cycles(I)$

petals are placed in the second quadrant so that their minimum points relative to $<_y$ are lined up parallel to the x-axis. The placement of the lines is slightly more complicated. It has to be done so that connectivity of two points by a line is not detectable in r moves of the $FO^r(\mathbb{P}, <_x, <_y)$-game. This is needed because connection information is not included in $cycles(I)$. In particular, the construction is dependent on r. For given r, we denote the homeomorphic representation of $cones(I)$ outlined above by $nice_r(cones(I))$.

Since all sentences in $FO^r_{top}(\mathbb{P}, <_x, <_y)$ are invariant with respect to homeomorphisms, the transformation does not affect equivalence with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$. Thus, it is sufficient to show that Duplicator has a winning strategy for the $FO^r(\mathbb{P}, <_x, <_y)$-game on $nice_r(cones(I))$ and $nice_r(cones(J))$. This is done by combining winning strategies from two games. The first is Duplicator's winning strategy for the FO^r-game on $cycles(I)$ and $cycles(J)$. The second is due to the non-definability of connectivity of two points by a line in $FO^r(\mathbb{P}, <_x, <_y)$ [GS97]. The combined strategy determines the cell where Duplicator picks the next point at every move. Moreover, the specific choice of points has to be done with care, because of the order relations. However, due to the similar shapes of $nice_r(cones(J))$ and $nice_r(cones(J))$, it is always possible to pick a point that sits in the desired order relative to previously picked points. Thus, $nice_r(cones(I))$ and $nice_r(cones(J))$ are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$, which completes the proof. □

Lemma 4.2 suggests a way to translate $FO^r_{top}(\mathbb{P}, <_x, <_y)$ sentences into FO_{inv} sentences. Indeed, it follows from the lemma that the set of invariants of instances satisfying an $FO^r_{top}(\mathbb{P}, <_x, <_y)$ sentence φ is a union of equivalence classes defined by the FO^r-equivalence of their cycles. Note that there are finitely many such equivalence classes. Thus, it suffices to show that one can effectively construct FO_{inv} sentences defining the equivalence classes of instances satisfying φ. At this point, Corollary 2.4 becomes very useful, since it provides a simple sufficient condition for FO^r-equivalence in terms of neighborhood types. By Corollary 2.4, two cycles are equivalent with respect to FO^r if they have the same (d, m)-type where $d = 3^r$ and $m = r3^r$. Note that for each (d, m)-type τ one can construct an FO formula ψ_τ stating that the cycles $cycles(I)$ of a given instance I has (d, m)-type τ. Also, given a (d, m)-type τ, we can check whether the instances I for which $cycles(I)$ has neighborhood type τ satisfy φ. This is done as follows:

(i) construct a cone instance I_τ whose corresponding cycle has type τ, if such exists (the brute-force approach takes 3-EXPSPACE in r; we can reduce the complexity to 3-EXPTIME using automata-theoretic techniques);

(ii) check if $I_\tau \models \varphi$.

If the answer to (ii) is positive, we say that τ *satisfies* φ. Let $\mathcal{T}$ consist of all sentences ψ_τ where τ is a (d, m)-type satisfying φ. The translation of φ into FO_{inv} is the sentence

$$inv(\varphi) \;=\; \bigvee_{\tau \in \mathcal{T}} \psi_\tau(\alpha_{cycles})$$

(Recall that α_{cycles} is the FO_{inv} formula defining $cycles(I)$ from $top(I)$). From the earlier remarks it follows that $\varphi(I) = inv(\varphi)(top(I))$ for every spatial instance I.

The above development, together with an analysis of the complexity of the construction, leads to the main result on the translation.

Theorem 4.3 *There exists a mapping inv from $FO_{top}(\mathbb{R}, <)$ sentences over* Reg *to FO_{inv} sentences over $inv(Reg)$ such that:*

1. *for each instance I over* Reg*, and $\varphi \in FO_{top}(\mathbb{R}, <)$,*

$$\varphi(I) = inv(\varphi)(top(I)).$$

2. *$inv(\varphi)$ is computable in* 3-EXPTIME *in the quantifier depth of φ.*

Remarks (i) Theorems 4.1 and 4.3 suggest an interesting trade-off between the expressive power of the target query language over the topological invariant and the complexity of the translation of $FO_{top}(\mathbb{R}, <)$ queries into the target language. Although all $FO_{top}(\mathbb{R}, <)$ queries can be uniformly translated into FO_{inv} queries, the complexity of the translation goes down from 3-EXPTIME to linear time if the target language is the more powerful $FO{+}\mu^+$ instead of FO_{inv}. (ii) We have shown that all $FO_{top}(\mathbb{R}, <)$ can be translated into FO_{inv} queries. One might naturally wonder if FO_{inv} and $FO_{top}(\mathbb{R}, <)$ express *precisely the same* topological properties. It is easily seen that this is not the case. For example, consider the instances in Figure 4. Clearly, FO_{inv} can distinguish between the two instances; it follows from [KPV97] that $FO_{top}(\mathbb{R}, <)$ cannot.

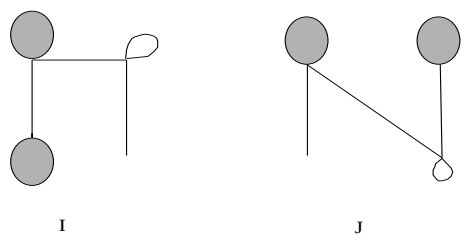

Figure 4: Instances I and J are distinguishable by FO_{inv} but not by $FO_{top}(\mathbb{R}, <)$

Practical considerations In what circumstances is translation a viable option? Suppose we have to evaluate an $FO_{top}(\mathbb{R}, <)$ query φ on a spatial instance I (over a single region name). We have to compare the following evaluation strategies:

(i) evaluate φ on I using classical quantifier elimination techniques; the complexity is PTIME in I and 2-EXPTIME in the quantifier depth of φ.

(ii) translate φ into $inv(\varphi) \in FO_{inv}$ and evaluate $inv(\varphi)$ on $top(I)$. The complexity is PTIME in $top(I)$ and[4] 3-EXPTIME in the quantifier depth of φ.

[4] A general purpose algorithm for evaluating $inv(\varphi)$ on $top(I)$ takes PSPACE in the size of $inv(\varphi)$, that is 3-EXPSPACE in the quantifier depth of φ. However, we can show a tailored evaluation algorithm that works in 3-EXPTIME in the quantifier depth of φ.

(iii) translate φ into $inv(\varphi) \in FO+\mu^+$ and evaluate $inv(\varphi)$ on $top(I)$. The complexity is PTIME in $top(I)$ and PSPACE in φ.

Clearly, the above information is not conclusive. How direct evaluation compares to translation into FO_{inv} or $FO+\mu^+$ followed by evaluation on the invariant depends critically on how much smaller $top(I)$ is compared to I. The gap can be arbitrarily large, so in principle translation can win. At this point it becomes useful to examine real data, in order to gauge how $top(I)$ compares to I in realistic settings. To this end, we examined cartographic data from the Sequoia 2000 project [Seq] and the French National Geographical Institute [IGN]. A first set from Sequoia 2000 contains cartographic data on ground occupancy in California (agricultural land, range land, forests, lakes, bays, estuaries, wetlands, beaches and tundra). The original data is represented by 31,778 polygons stored as a list of 2,557,071 points taking 20 bytes each. The corresponding topological invariant has 190,045 cells taking 3 bytes each, which is 1/90 of the original data. A second set from Sequoia 2000 contains data on rivers, lakes and estuaries and contains 3410 polygons with 135,527 points at 20 bytes each. The invariant has 4,570 cells at 2 bytes each, which is 1/300 of the raw data. Data from the French National Geographical Institute contains cartographic data on the city of Orange (France) and surroundings. It consists of 145 polygons with 11,916 points at 18 bytes each. The topological invariant has 1,487 cells at 2 bytes each (1/72 of the original data).

As it turns out, other characteristics of data may also make translation viable. In cartographic data, it is reasonable to assume that there is a constant bound on the number of lines intersecting at the same point. In the data sets mentioned above, the average number of lines intersecting at a point is 4.5 for both, with maxima of 12 for Sequoia 2000 and 8 for the French National Geographical Institute data. If a constant bound is assumed, then the overall complexity of option (ii) (evaluation via translation to FO_{inv}) goes down to polynomial in $top(I)$ and 2-EXPTIME in the quantifier depth of φ. This wins over direct evaluation, which is polynomial in I rather than $top(I)$. The same complexity is obtained in the case of *fully two-dimensional*[5] regions. These are regions equal to the closure of their interior; in other words, the only edges and vertices occur on boundaries of faces.

We examined two possibilities for the targets of the translation of $FO_{top}(\mathbb{R}, <)$ queries: (ii) FO_{inv} and (iii) $FO+\mu^+$. On the face of it, the complexity of the translation and overall query evaluation seems to favor (iii). However, there is a caveat: the stated complexities of the translation and evaluation hide large constants. Nonetheless, (iii) seems likely to win over (ii), assuming that a recursive query evaluation engine evaluating the *fixpoint* queries on the invariant is available.

Lastly, another option of practical interest is to avoid query translation altogether by using the following strategy for evaluating φ on input I:

(iv) construct a standard linear instance I' such that $top(I') = top(I)$ and evaluate φ on I'.

The instance I' can be evaluated from $top(I)$ in PTIME, by Theorem 2.2. Altogether, I' can be constructed from I in PTIME. Note that the size of I' is roughly the same as the size of $top(I)$. Thus, maintaining the simplified linear instance I' as an annotation instead of $top(I)$ combines the advantage of avoiding query translation with evaluation on a smaller input. Note however that the only algorithm known at this point for constructing I' is via the topological invariant.

[5] Term introduced by B. Kuijpers and J. Van Den Bussche.

5 Conclusions

We examined the use of topological annotations to evaluate topological queries against a spatial database. The first main result, showing that *fixpoint* expresses exactly the PTIME queries on topological invariants, shows that topological invariants are especially well-behaved with respect to descriptive complexity, and provides an appealing target language for the translation of spatial queries into queries against the invariant. If recursion is not supported, $FO_{top}(\mathbb{R}, <)$ queries can be translated into FO_{inv} queries in the case of one-region schemas. The general case of multiple regions remains open. The complexity of the translation into FO_{inv} is high, but so are the potential savings due to the difference in size between spatial databases and their topological invariants. Other characteristics of some data sets, such as the existence of a small bound on number of lines intersecting at one point, also favor translation. Finally, another appealing option is to use as annotation a linear embedding of the topological invariant, which circumvents the need for query translation.

It is natural to wonder if these techniques can be extended beyond dimension two. This question is examined in [Seg], and the picture is largely negative. The existence of a topological invariant for 3-dimensional semialgebraic databases implies a positive answer to an open problem in knot theory: the existence of an invariant for topologically equivalent knots [Cro63]. In dimension four (and higher), it is shown in [Seg] that there is no finite topological invariant, because topological equivalence itself is undecidable. This is shown by adapting the proof of an undecidability result on topological equivalence of manifolds [Mar58]. The latter proof is by reduction of the word problem for finitely generated groups to isomorphism of the fundamental groups of two topological spaces, which in turn is equivalent to their being homeomorphic.

References

[AHV95] S. Abiteboul, R. Hull and V. Vianu. Foundations of Databases. Addison-Wesley, 1995.

[AV91] S. Abiteboul and V. Vianu. Datalog Extensions for Database Queries and Updates. *Journal of Computer and Systems Science*, 43,1, 1991.

[AV95] S. Abiteboul and V. Vianu. Computing with first-order logic. *Journal of Computer and System Sciences* 50:309–335, 1995.

[BDLW96] M. Benedikt, G. Dong, L. Libkin and L. Wong. Relational expressive power of constraint query languages. In Proc. ACM Symp. on Principles of Database Systems, 1996.

[CFI92] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound onb the number of variables for graph identification. *Combinatorica* 12:389–410, 1992.

[Cor79] J.P. Corbett. Topological principles in cartography. Technical Paper 48, US Bureau of the Census, Washington D.C., 1979.

[Cro63] R. Crowell and R. Fox. Introduction to knot theory. Springer-Verlag, GTM vol. 57, 1963.

[DLW95] A. Dawar, S. Lindell, and S. Weinstein. Infinitary logic and inductive definability over finite structures. *Information and Computation* 119:160–175, 1995.

[FSV95] R. Fagin, L. Stockmeyer and M. Vardi. On Monadic NP vs. Monadic co-NP. In *Inf. and Computation*, **120(1)**, pp. 78-92, 1995.

[Gro97] M. Grohe. Large Finite Structures with Few L^k-Types. In *Proc. Symp. on Logic in Computer Science*, 1997.

[GO93] E. Grädel and M. Otto. Inductive definability with counting on finite structures. In E. Börger et al., editors, *Computer Science Logic, CSL'92 Selected Papers*, LNCS vol.702, pp. 231–247, Springer-Verlag, 1993.

[GK97] S. Grumbach and G. Kuper. Tractable recursion over geometric data. In *Proc. Int'l. Conf. on Constraint Programming*, 1997.

[GS97] S. Grumbach and J. Su. Queries with Arithmetical Constraints. *Theoretical Computer Science*, **173**, pp. 151-181, 1997.

[GS98] S. Grumbach and J. Su. Finitely representable databases. *Journal of Computer and System Sciences*, to appear.

[GST94] S. Grumbach, J. Su and C. Tollu. Linear constraint databases. In *Proc.Logic and Complexity Workshop*, D. Leivant ed., Indiana, 1994.

[Gur84] Y. Gurevich. Toward logic tailored for computational complexity. In M. M. Richter et al., editor, *Computation and Proof Theory, Lecture Notes in Mathematics 1104*, pages 175–216. Springer-Verlag, 1984.

[IGN] La Base de Données Topographiques de l'I.G.N.. (The Cartographical Databse of the National Geographic Institute, in French.) Bulletin d'Information de l'I.G.N., No. 59, 1991.

[I86] N. Immerman. Relational queries computable in polynomial time. Information and Control 68 (1986), pp. 86-104.

[I87] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proc. 2nd Structure in Complexity Conference*, pp. 194–202, 1987.

[KKR90] P.C. Kanellakis, G.M. Kuper and P.Z. Revesz. Constraint query languages. In Proc. ACM Symp. on Principles of Database Systems, 1990, 299–313.

[KY85] D. Kozen and C-K. Yap. Algebraic Cell Decomposition in *NC*. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1985, 515-521.

[KPV95] B. Kuijpers, J. Paredaens and J. Van den Bussche. Lossless Representation of Topological Spatial Data. In *Proc. Fourth Int'l. Symp. on Large Spatial Databases*, 1995.

[KPV97] B. Kuijpers, J. Paredaens and J. Van den Bussche. On topological elementary equivalence of spatial databases. In *Proc. Int'l. Conf. on database Theory*, 1997, pp.432-446.

[M85] S. Morehouse. ARC/INFO: A Geo-Relational Model for Spatial Information. In *Proc. Int'l. Symp. on Computer Assisted Cartography (Auto-Carto 7)*, pp. 388-397, 1985.

[M89] S. Morehouse. The Architecture of ARC/INFO. In *Proc. Int'l. Symp. on Computer Assisted Cartography (Auto-Carto 9)*, pp. 266-277, 1989.

[Mar58] A.A. Markov. Unsolvability of the problem of homeomorphy. In *Proc. Int'l Congress of Mathematics*, 1958, pp. 300-306. In Russian.

[PSV96] C.H. Papadimitriou, D. Suciu and V. Vianu. Topological Queries in Spatial Databases. In Proc. ACM Symp. on Principles of Database Systems, 1996.

[Par+94] J. Paredaens, J. Van Den Bussche and D. Van Gucht. Towards a theory of spatial database queries. In *Proc. ACM Symp. on Principles of Database Systems*, 1994, 279–288.

[Par95] J. Paredaens. Spatial Databases, the Final Frontier. In *Proc. 5th Int'l. Conf. on Database Theory*, 1995, 14–32.

[Par+95] J. Paredaens, J. Van Den Bussche and D. Van Gucht. First-order queries on finite structures over the reals. In *Proc. Conf. on Logic in Computer Science*, 1995.

[Seg] L. Segoufin. PhD Thesis, in preparation.

[Seq] M. Stonebraker, J. Frew, K. Gardels and J. Meredith. The Sequoia 2000 Storage Benchmark. In *Proc. ACM SIGMOD Int'l. Conf. on the Management of Data*, 1993 .

[Tar51] A. Tarski. A Decision Method for Elementary Algebra and Geometry. University of California Press, 1951.

[V82] M.Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, 1982, pp. 137-146.