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

December 7, 1999

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 + counting*, *fixpoint*, and first-order on the topological invariant side. In particular, it is shown that *fixpoint + counting* expresses precisely all the PTIME queries on topological invariants; if the regions are connected, *fixpoint* expresses all 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 are 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 "the point p is North of the point q" is *not* topological. In previous work [PSV99] 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 smaller in size than the original database, 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:

*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.

- 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 + counting* expresses precisely the PTIME queries over topological invariants (*fixpoint* alone is sufficient if the regions are connected). This should be contrasted with the situation on arbitrary structures, where *fixpoint + counting* 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 + counting* (or *fixpoint* in the case of connected regions) 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.

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 + counting* 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 + counting* 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. Unfortunately, the translation cannot be extended to the multi-region case [GroSeg99].

Interestingly, even when both FO and *fixpoint + counting* 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 hyperexponential 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 + counting*. 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.

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.

Various topological invariants have been used in geographic information systems in order to speed up the evaluation of queries that make reference to topological properties of regions (e.g. in the ARC/INFO system [Mor85, Mor89]). Among them, the 4-intersection model of topological relationships [FK86, Her91, Wor92] has been widely adopted in geographic information systems and has been used in several spatial query languages [Ege94, OV91, SZ91]. The satisfiability problem for 4-intersection relationships is investigated in [GPP95]. The expressiveness of these relationships has been investigated by Egenhofer and Franzosa [EF91], who make the argument that they are natural and cognitively plausible, and observe that they cover all possibilities that are expressible in the language that includes disjointness of two sets, interior, exterior, boundary, and Boolean connectives. The 4-intersection invariant was further refined by Egenhofer and Franzosa, by taking into account additional information, such as the number and dimension of components of the boundary intersection of two regions [FE92, Ege93, EF95a]. In particular, the invariant exhibited in [EF95a] is claimed (without proof) to completely characterize two regions (discs) up to homeomorphism.

Unlike the above topological invariants, the topological invariants we use are *lossless*, i.e. they completely characterize the topology of a set of regions. The invariants contain information similar to the PLA model proposed by the U.S. Census Bureau, which provides topological properties on points, lines, and areas [Cor79, Par95]. The invariants we use can be viewed as an augmentation of the PLA model. Models using decompositions of the space into cell complexes have been used in the geographic information systems community for some time (e.g., see [FK86, Her91, Wor92]). The complexity of computing topological information based on cell complexes, similar to the PLA model and to our invariants, has been studied in computational geometry [BKR86, KY85]. While a flaw has been discovered in the complexity analysis of [BKR86], later modifications recovered, and even improved upon their upper bounds, see Renegar [Ren92].

The topological invariant we use is essentially the one introduced in [PSV99]. The invariants proposed in [KPV95] are also close to the topological invariants we consider, but capture isotopy-generic information.

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 [KKR95]. 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].

Our result that *fixpoint + counting* expresses PTIME on topological invariants is closely related to an elegant result in finite-model theory independently obtained by Grohe [Gro98]. It is shown there that *fixpoint + counting* expresses PTIME on planar graphs, and *fixpoint* alone captures PTIME on 3-connected planar graphs.

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 + counting* on topological invariants, and related results. Section 4 presents the results on the translation of spatial topological $FO(\mathbb{R},<)$ queries into *fixpoint + counting* and FO queries on topological invariants.

2 Preliminaries

Practical spatial databases (such as geographic information 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 *Reg* 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 (bounded and closed), and specified by a disjunction of conjunctions of polynomial inequalities with rational coefficients (this later property is usually referred to as *semi-algebraic*). In this paper, we will call *connected* a region which boundary is connected. 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 τ 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 satisfies τ iff J satisfies τ.

Topological invariants In [PSV99] 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 [PSV99] 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 [PSV99] pertaining to the topological invariant are easily adapted to the model used here. We briefly describe the construction 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 I is globally invariant by λ ($\lambda(I) = I$) then, for every cell c, $\lambda(c)$ is a cell. It can be 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[2] on *cell complexes* obtained in [BKR86, 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 $\mathbb{R}^2 - \{\text{a finite set of points}\}$. 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][3] that a cell complex can be constructed from a given semi-algebraic spatial instance in NC. One can further show that the maximum (see definition above) 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 built 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. *Edge-Vertex* is a binary 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 5-ary relation providing the clockwise and counterclockwise orientation of edges incident to each vertex. More precisely, $(\hookleftarrow, v, e_1, e_2, e_3) \in Orientation$ iff v is a vertex, e_1, e_2, e_3 are edges of faces incident to v, and e_2 lies between e_1 and e_3 in the clockwise order on the incidents cells of v, and $(\hookrightarrow, v, e_1, e_2, e_3) \in Orientation$ iff v is a vertex, e_1, e_2, e_3 are cells incident to v, and e_2 lies between e_1 and e_3 in the counterclockwise order on the incident cells of 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*).

Remark: The topological invariant of [PSV99] had a slightly different schema and definition. The *Orientation* relation was 4-ary instead of 5-ary, and corresponded to the successor relation of the incident cells of a given vertex v. For technical reasons, in our setting *Orientation* provides the full cyclic order of the cells incident to v (without this modification, Theorem 4.9 does not hold). This change does not affect the results on the invariant from [PSV99].

The main result on the topological invariant is the following.

[2]As discussed earlier, while a flaw has been discovered in the complexity analysis of [BKR86], later modifications recovered, and even improved upon their upper bounds, see Renegar [Ren92].

[3]Actually [KY85] need to do a linear change of the coordinate system in order to be able to compute a cell complex. As this doesn't affect the topological properties this doesn't affect the results here. See [BCR98, Cos89] for a discussion on cell complex.

Theorem 2.1 *[PSV99] 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 *[PSV99] 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 [Imm86, Var82] and *while* expresses exactly PSPACE [Var82] *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). Queries such as parity can be expressed if a counting mechanism is added to FO+μ^+, yielding the *fixpoint + counting* queries. The counting is provided by counting quantifiers of the form $\exists^i x\ \varphi(x)$ where integer variables i range over an ordered domain of the same size as the input finite domain and disjoint from it (see [CFI92, GO93] for details). Unfortunately, *fixpoint + counting* still fails to express PTIME [CFI92].

Most of the languages previously proposed for spatial databases, including constraint query languages [KKR95, GS99, GST94, Par+95, BDLW98], 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 [KKR95] 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 $\exists x\ P(x, x)$[4] However, it is shown in [PSV99] (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[5]. In this paper we assume 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-Fraïssé 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 $qd(\varphi) = 0$

[4]It is shown in [PSV99] that all queries of $FO(\mathbb{P}, <_x, <_y)$ are invariant with respect to the group of *monotone isomorphisms* $\mathcal{M} = \{\lambda \mid \lambda((x, y)) = (\rho_1(x), \rho_2(y))\}$ with $\rho_1, \rho_2 : \mathbb{R} \to \mathbb{R}$ bijective and increasing. Since the query $\exists x\ P(x, x)$ is not invariant with respect to $\mathcal{M}$, it cannot be expressed in $FO(\mathbb{P}, <_x, <_y)$.

[5]The undecidability is shown by a straighforward reduction of satisfiability for $FO(\mathbb{R}, <)$, known to be undecidable [GS99]

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 I, J are FO^r-equivalent iff Duplicator has a winning strategy for the FO^r-game on I and J. See [EF95b] for more details.

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 $FO^r(\mathbb{P}, <_x, <_y)$.

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 + counting* queries capture precisely PTIME over this class of structures. If we restrict our attention to instances were the regions are connected (i.e. have a connected boundary) then it can be shown that *fixpoint* queries capture precisely PTIME. This positive result should be contrasted with the situation on arbitrary structures, where there is a large gap between *fixpoint* (+ *counting*) and PTIME (in the absence of order). Moreover, it is conjectured that there is no language capturing PTIME on arbitrary structures [Gur88].

Fixpoint on connected regions

We first consider the case when all regions are connected and show that *fixpoint* expresses PTIME on topological invariants of such instances. Then we consider the general case (where regions are not necessarily connected) and show that *fixpoint + counting* expresses PTIME on topological invariants.

In the remainder of this section we assume all regions are connected, unless otherwise stated. The proof that *fixpoint* captures PTIME on topological invariants in the connected case is conceptually easy, but requires some careful development. We use the classical result that *fixpoint* captures PTIME on ordered structures [Imm86, Var82]. 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 by definition, 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 are the elements of the relation Vertice and Edge of the invariant, and whose edges correspond to the relation *Edge-Vertex* in the invariant. A *connected component* of the topological invariant is a connected component of the skeleton of the invariant. To illustrate this definition, the connected components in Figure 1 are $c_1, \cdots, c_7$. Since all regions are connected, 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.

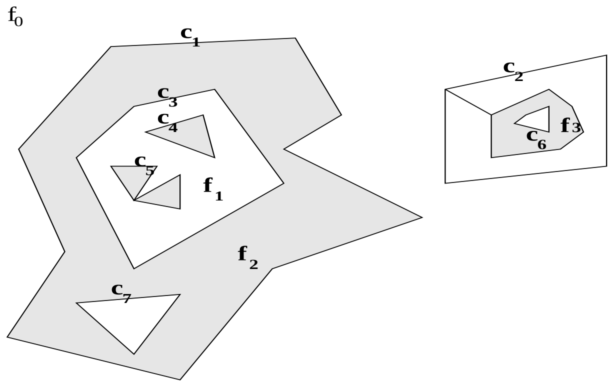

Figure 1: A spatial instance.

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 $f_i, 0 \leq i \leq k$). For instance, in Figure 1, the connected components c_1 and c_2 are at distance 0 from f_O, $c_3, c7$ at distance 1, and c_4, c_5 and c_6 at distance 2. 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. In Figure 1 the boundary of region f_2 touch three connected components, c_1, c_3, c_7, and is therefore associated to c_1 because c_1 is at distance 0 from f_0 while the other two are at distance 1. It is a straightforward exercise to see that the previous construction 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 associated faces of the connected component of v*.*

Proof For the moment, suppose that all connected components have at least one proper edge. We describe the construction of the $FO + \mu^+$ query $\alpha(\omega, v, e, x, y)$ computing the orders $\leq_{\omega,v,e}$ for each connected component $\mathcal{C}$ containing vertex v with an adjacent proper edge e, and fixed orientation ω. The query first constructs the set of tuples $< \omega, v, e >$ where $\omega \in \{\hookleftarrow, \hookrightarrow\}$, v is a vertex in $\mathcal{C}$ and e is a proper edge adjacent to v (this is an FO query, given $\mathcal{C}$). For each such tuple $< \omega, v, e >$, the following is done. First, an order on the vertices of $\mathcal{C}$ is defined; simultaneously, to each vertex is associated a particular proper edge adjacent to it. Then the order is extended to the edges and faces of $\mathcal{C}$. The order on vertices is defined as follows. The first vertex is v. Suppose $e_1, \ldots, e_k$ are the proper edges adjacent to v in order of the ω-orientation, starting from e, and let $v_1, \ldots, v_k$ be their respective endpoints other than v. The first iteration computes the partial order $v, v_1, \ldots, v_k$. At the same time, the proper edge associated to v_i is defined to be e_i, $1 \leq i \leq k$. This is iterated starting from the tuples ω, v_i, e_i, eliminating repetitions of vertices. The partial order obtained for the pair v_i, e_i is then inserted between v_i and v_{i+1}, for $1 \leq i < k$ and that for v_k, e_k is inserted following v_k. Clearly, this can be achieved by a *fixpoint* query which produces a total order on all vertices of $\mathcal{C}$, and associates along the way a proper adjacent edge to each vertex. Now the ordering is extended to edges as follows. Edges (proper or not) are first ordered in lexicographic order of the ranks of their endpoints (the edges are allready ordered). Edges with the same endpoints $u \leq v$ are ordered in the ω-orientation relative to the proper edge associated with u. Consider two faces f, f' in $\mathcal{C}$ (recall the earlier definition of a face being in a connected component). It is easily seen that no two faces in $\mathcal{C}$ have the same set of edges in the intersection of their boundary with $\mathcal{C}$. Since edges are already ordered, the faces of $\mathcal{C}$ can be ordered using their associated set of edges in $\mathcal{C}$. Thus, vertices, edges and faces of $\mathcal{C}$ are ordered. A total order on the universe of $\mathcal{C}$ is obtained by having all vertices be smaller than all edges which in turn are smaller than all faces in $\mathcal{C}$.

Lastly, we briefly discuss the special case of connected components with no proper edges. There are three cases:

1. The connected component consists of a single isolated vertex.
2. The connected component consists of a single edge e (a loop with no endpoints).
3. The connected component consists of a vertex and a set of loops around the vertex.

Case (1) is trivial: the order consists of the pair $\{< v, v >\}$, where v is the vertex constituting the connected component. Although the order is uniquely determined without fixing any elements, for the sake of uniformity we make the order dependent on the pair $< \omega, v, v >$ (this keeps the result of α 5-ary). Case (2) is similar. For case (3), we consider pairs v, e where v is the unique vertex and e is a loop adjacent to the exterior cell. The order on vertices is simply $< v, v >$. The order on edges is determined by the ω-orientation of edges starting from e. There is one difficulty: loops occur twice in ω-orientation order. Therefore, the successor of e is ambiguous: however, there is a single loop which follows e in the ω-orientation and is also adjacent to the exterior face. This is taken to be the successor of e. Subsequent successors are uniquely determined (each edge is inserted in the ordering only when first encountered in the ω-orientation). If e is the only loop around v then the order is simply $< e, e >$. The faces of the connected component are ordered as described earlier. $\square$

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:

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

where

$$\mathit{Is\text{-}Orientation}(x) = \exists v \exists e_1 \exists e_2 \exists e_3 \mathit{Orientation}(x, v, e_1, e_2, e_3).$$

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 by definition, 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 of instances with connected regions.*

Proof We have seen that the parameterized orderings $\leq_\pi$ can be defined by a *fixpoint* query. Let us denote by ξ_π the parametric variables of $\leq_\pi$ (all free variables except x and y in the fromula defining $\leq_\pi$) . With $\leq_\pi$ available the resulting fixpoint query is obtained by a case analysis over the set of partitions π of *Reg*:

$$\bigvee_\pi \exists \xi_\pi [\sigma_\pi \wedge \varphi_\pi(\leq_\pi)].$$

□

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 of spatial instances with connected regions.*

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 involves 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$^\neg$.

Fixpoint + *counting* on arbitrary invariants

Theorem 3.2 relies crucially on the assumption that regions are connected. In the case where this is not required, each region may have 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* expresses precisely the PTIME queries on topological invariants. 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 [Imm86, Var82].

Theorem 3.4 Fixpoint + counting *expresses exactly the* PTIME *queries on topological invariants.*

Proof As stated above, we will exhibit a *fixpoint + counting* query which, given as input a topological invariant I, produces an isomorphic copy of I on the auxiliary ordered domain. Clearly this is sufficient to establish the theorem, since *fixpoint* expresses PTIME on ordered structures.

The proof involves several stages. First recall that, using Lemma 3.1, we can deal with single connected components: by the lemma, a *fixpoint* query can define polynomially many orderings of the invariant. Each ordering $<$ is parameterized by a tuple of fixed arity. For each fixed ordering $<$, the invariant can be viewed as an ordered structure $c_<$. Note however that different orderings produce different ordered structures. To produce a single isomorphic copy on the auxiliary ordered domain, it is sufficient to define some standard lexicographic ordering among ordered structures $c_<$ and pick the minimum ordered structure c_{min} with respect to the lexicographic order. It is now easy to construct an instance isomorphic to c_{min} over the auxiliary ordered domain.

The idea just described can be extended to multiple connected components. First, we define a labeled tree that captures the way connected components are embedded into faces. The nodes of the tree are (conceptually) the connected components of the invariant (except for the root, denoted ∞). Each connected component c is in fact represented in the tree by its set of vertices[6] V_c. The edges of the tree are labeled by faces (into which the children connected components are embedded). The tree is defined inductively as follows. The outgoing edges from the root ∞ are all labeled f_0 (the external face). The children of the root are all connected components which share boundaries with f_0. Now suppose c is a node of the tree. The outgoing edges from c are labeled by all faces f that share a boundary with c and do not yet occur as labels in the tree. There is an edge labeled f from c to each connected component c' other than c that shares boundaries with f. It is easy to see that the tree just described can be defined by a *fixpoint* query. We refer to this tree as the *connected component tree* of the invariant, denoted T_∞. For example, the connected component tree corresponding to the instance in Figure 1 is represented in Figure 2.

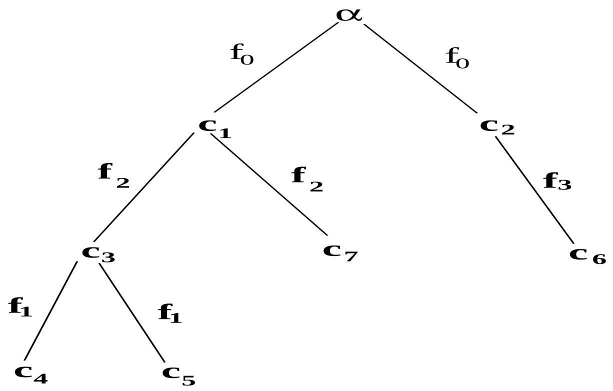

Figure 2: Connected component tree for instance in Figure 1.

For each node c in T_∞, let T_c be the subtree of T_∞ with root c. Note that each T_c corresponds to an invariant comprising c and the connected components embedded in c, denoted $inv(T_c)$. We show the following:

(†) There exists a *fixpoint + counting* query that defines, for each subtree T_c of T_∞, a structure $V_c \times inv_N(T_c)$ where $inv_N(T_c)$ is an isomorphic copy of $inv(T_c)$ on the auxiliary ordered domain.

In particular, (†) shows that one can define in *fixpoint + counting* a structure $\{\infty\} \times inv_N(T_\infty)$, and therefore $inv_N(T_\infty)$. This is an isomorphic copy of the original invariant over the auxiliary ordered domain.

To prove (†) we proceed inductively by the depth of the subtree. For the basis, consider a subtree of depth zero consisting of a single node c (one connected component). As discussed earlier, this case can be

[6] For simplicity of exposition, we omit the degenerate case of connected components without vertices.

dealt with using the parameterized orders definable in *fixpoint* by Lemma 3.1. Now consider a subtree T_c of depth > 0. By induction, suppose that for each child d of c, a structure $V_d \times inv_N(T_d)$ has been defined, where V_d is the set of vertices of d and $inv_N(T_d)$ is an isomorphic copy of $inv(T_d)$ over the auxiliary ordered domain. To extend the construction to T_c, we first define parameterized orders over $inv(c)$ using the *fixpoint* query provided by Lemma 3.1. Consider one such order $<$. The cells of $inv(c)$ are totally ordered by $<$ and so $inv(c)$ can easily be mapped to an isomorphic structure $inv_N(c)$ over the auxiliary ordered domain. Now we wish to extend this mapping to $inv(T_c)$. Note that children of c embedded in distinct faces of c can be ordered according to the order on faces induced by $<$. For children d, d' of c embedded in the same face of c there are two cases. First, if $inv_N(T_d) \neq inv_N(T_{d'})$, then d and d' can be ordered in the lexicographic order of $inv_N(T_d)$ and $inv_N(T_{d'})$. If $inv_N(T_d) = inv_N(T_{d'})$ then T_d and T'_d are isomorphic and cannot be ordered. However, this difficulty can be overcome using counting. Indeed, it is easy to see that one can define in *fixpoint + counting* , for each child d of c, the number n of children of c that are embedded in the same face as d and are isomorphic to it. Then one can simply produce n disjoint isomorphic copies of d on the auxiliary ordered domain. In summary, one can define in *fixpoint + counting* one isomorphic copy $inv_N(T_c)$ over the auxiliary ordered domain for each ordering $<$ previously defined on $inv(c)$. To produce a single copy $inv_N(T_c)$ is is enough to pick among the copies corresponding to the different orderings the lexicographically minimum one. Lastly, this structure is associated with c by taking the product of V_c with $inv_N(T_c)$. This completes the induction and proves (†) and the theorem. □

The above result can also be extended to language subsuming *fixpoint*, such as *while*. We denote the natural extension of the partial fixpoint queries with counting by *while + counting*.

Corollary 3.5 While + counting *expresses exactly the* PSPACE *queries on topological invariants.*

Remark Theorems 3.4 and 3.2 crucially rely on the fact that regions are part of the database schema and so are fixed. Now suppose that this is relaxed so that region names are part of the instance rather than 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 + counting* on topological invariants has consequences for the capture of the PTIME *topological queries on spatial instances* (recall that such queries are assumed to be Boolean). 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 + counting* 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 semi-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 + counting* 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. Note that *fixpoint + counting* can be replaced by *fixpoint* if regions are connected.

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 non-topological 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 + counting* on topological invariants provides for free an effective translation of topological $FO(\mathbb{R},<)$ queries into *fixpoint + counting* queries on the invariant (*fixpoint* suffices if the regions are connected). 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 + counting* (*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* fixpoint + counting *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. Using a *fixpoint + counting* query, define an isomorphic copy T_N of T on the auxiliary ordered domain (as in the proof of Theorem 3.4).
2. Construct from T_N a semi-linear instance J over Reg such that $top(J) = T$.
3. Evaluate φ on J and output the result (which recall is Boolean).

We claim that (1)-(3) can be simulated by a *fixpoint + counting* query $inv(\varphi)$ constructible from φ in linear time. First, note that the *fixpoint + counting* query in (1) is fixed and can be constructed in constant time with respect to φ. Consider (2)-(3). By Theorem 2.2, (2) can be done in PTIME, and (3) can be done in PTIME by [KKR95]. Thus, steps (2) and (3) combined take PTIME with respect to T_N. Thus, there is a PTIME Turing machine M_φ that implements (2) and (3). Essentially, this is a Turing machine that performs step (2) (this is fixed and independent of φ), composed with a universal Turing machine for evaluating $FO(\mathbb{R},<)$ formulas, to which φ is provided as parameter. Since the universal component is fixed it is clear that M_φ can be constructed in time linear with respect to φ. Next, since T_N is ordered and *fixpoint* can simulate all PTIME Turing machines on ordered domains, there exists a *fixpoint* query $\psi(M_\varphi)$ that simulates M_φ on input T_N. The standard construction used in the simulation of PTIME Turing machines by *fixpoint* produces a *fixpoint* formula whose size is linear in the size of the Turing machine (e.g., see [AHV95]). Thus, $\psi(M_\varphi)$ can be constructed in linear time with respect to M_φ and therefore with respect to φ. Altogether, the composition of the *fixpoint + counting* query in (1) with the *fixpoint* query $\psi(M_\varphi)$ can be constructed in time linear in φ. □

If the regions are connected we can similarly show:

Theorem 4.2 *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 *with connected regions,* $\varphi(I) = inv(\varphi)(top(I))$.

2. $inv(\varphi)$ *is computable in linear time in the size of* φ.

Remark 4.3 *In the above, it is assumed that the queries in* $FO(\mathbb{R}, <)$ *to be translated are known to be topological. As discussed in Preliminaries, we do not deal here with the issue of verifying or enforcing that the input* φ *to the translation 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. If the input* φ *is not topological, the result of the translation is no longer equivalent to* φ. *Instead, it is equivalent to the "topological closure" of* φ, *denoted* φ_{top}, *defined as follows:*

$$\varphi_{top} = \{I \mid \exists J \; I \; \equiv_{\mathcal{H}} \; J, \; J \models \varphi\}.$$

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[7]:

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

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

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 non-topological statements involving reals and $<$. We solve the translation problem for the case when Reg contains a single region name; it has been recently shown that the translation is not possible with several region names [GroSeg99]. Recall however that single regions can be arbitrarily complicated (as long as they are semi-algebraic 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 develop the technical machinery needed for 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 [PSV99], 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 $FO_{top}(\mathbb{P}, <_x, <_y)$, see Preliminaries) 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 3, the instance $cones(I)$ is represented in Figure 4.

Remark 4.4 *The normal form of [KPV97] is introduced as a technical tool in the proof of their Proposition 10. It is referred to as* an instance with flowers and stems. *The normal form is not unique: different instances in normal form have the same* flowers *but these can be conected in different ways by the* stems. *Since the differences are irrelevant for our purpose, we denote with* $cones(I)$, *by abuse of notation,* one *of the possible instances in normal form constructed from* I.

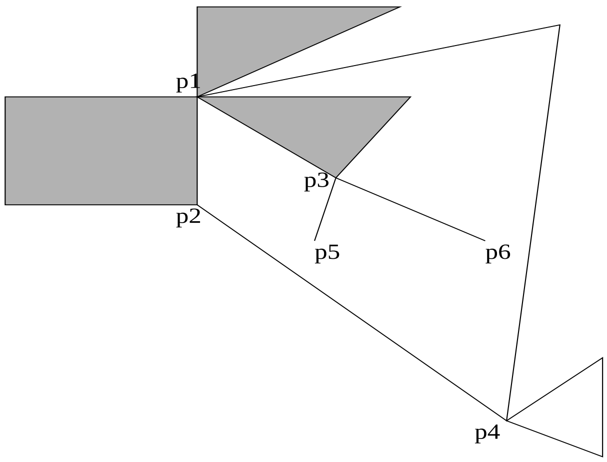

Figure 3: An instance I

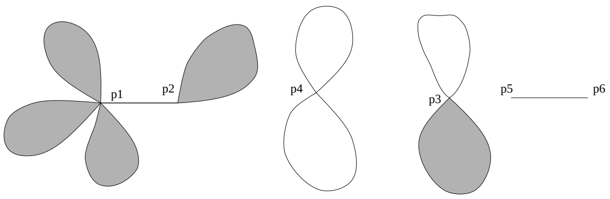

Figure 4: An instance $cones(I)$

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: to each cone corresponds two cycles representing the list of edges and faces adjacent to the vertex of the cone in clockwise, resp. counterclockwise order. More precisely, consider the structure $cycles(I)$ consisting of the following relations:

- the 4-ary relation *Between* such that *Between*(ω,x,y,z) holds if x, y and z are cells adjacent to some vertex, and y lies between x and z in the orientation ω.
- the restriction of relations *Edge* and *Face* of $top(I)$ to elements occurring in relation *Between*.

Clearly, the structure $cycles(I)$ is essentially a set of 2-colored cycles (the nodes are colored according to whether they are a face or an edge). For example, $cycles(I)$ for the instance I in Figure 3 is represented in Figure 5. In the figure, a dark node represents a face in the region, an empty node represents an edge.

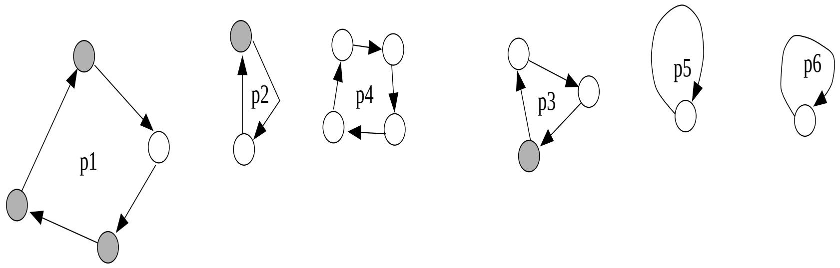

Figure 5: $cycles(I)$, clockwise orientation

[7]With slight abuse of notation.

Lemma 4.5 *The structure cycles(I) can be defined by first-order means from top(I). Let α_{cycles} be an FO_{inv} formula defining cycles(I) from top(I).*

The proof of this lemma is rather straightforward. Each element of *Vertex* in *top*(I) defines a cycle in *cycle*(I) which element are its neighborhoods defined by the relation *Edge-Vertex* and *Face-Edge*, ordered using the *Orientation* predicate.

The key observation in the translation provides a condition on *cycles*(I) and *cycles*(J) that ensures $FO^r_{top}(\mathbb{P}, <_x, <_y)$-equivalence of *cones*$(I)$ and *cones*(J). We first consider the case when *cycles*(I) and *cycles*(J) consist of single cycles (in the two orientations), then consider the case of multiple cycles. We first prove the following.

Lemma 4.6 *Let cones(I), cones(J) be spatial instances consisting of a single cone. If cycles(I) and cycles(J) are equivalent with respect to FO^{r+2} then cones(I) and cones(J) are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$.*

Proof Suppose *cones*(I), *cones*(J) consist of single cones and *cycles*(I) and *cycles*(J) are equivalent with respect to FO^{r+2}. Note that each of *cycles*(I) and *cycles*(J) consists of a single cycle (in the two orientations). We wish to prove that *cones*(I) and *cones*(J) are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$. The difficulty is that *cones*(I) and *cones*(J) are generally *not* equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$. To show their equivalence with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$, we construct instances *nice*(*cones*(I)) and *nice*(*cones*(J)) such that:

- *nice*(*cones*(I)) is equivalent to *cones*(I) with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$;
- *nice*(*cones*(J)) is equivalent to *cones*(J) with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$; and,
- *nice*(*cones*(I)) and *nice*(*cones*(J)) are equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$.

To construct *nice*(*cones*(I)) from *cones*(I) (and similarly for J) we modify *cones*(I) in two stages. First, we apply a homeomorphism to *cones*(I). In the second stage we replace the resulting instance with an instance equivalent to it with respect to $FO_{top}(\mathbb{P}, <_x, <_y)$. Since both transformations preserve equivalence with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$, *nice*(*cones*$(I)$) is equivalent to *cones*(I) with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$. Lastly, we show that *nice*(*cones*(I)) and *nice*(*cones*(J)) are equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$ (and therefore with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$). The above sequence of equivalences shows that *cones*(I) and *cones*(J) are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$.

We now describe the two stages in the construction of *nice*(*cones*(I)) and *nice*(*cones*(J)). One subtlety is that the constructions for I and J are not independent, as will become clear shortly. The purpose of the first stage in the construction of *nice*(*cones*(I)) and *nice*(*cones*(J)) is to lie out the lines and petals (faces) of the two cones in a very regular manner, which will eventually facilitate playing an $FO^r(\mathbb{P}, <_x, <_y)$-game on the modified instances. In the first stage, the vertex of each cone is placed at the origin, and all petals are placed in the second quadrant so that their minimum points relative to $<_y$ are lined up parallel to the x-axis (see Zone A in Figure 6). Consider a bounding box that includes all the petals of the cone. The lines are *pushed away* to a region that lies to the lower right of the bounding box, and the connections between the lines are drawn there (Zone B in Figure 6).

Let us focus on Zone B of the instances constructed so far. Note that the pattern of connections in Zone B can be quite complex. However, the connections can be drastically simplified as illustrated in Figure 7: lines are simply connected in consecutive pairs (there is always a even number of lines because each line enters and leaves the zone exactly once). Consider the subinstance consisting of a bounding box around Zone B, where the connections occur. Each line intersects the left edge of the box at a point, and connections between the lines can be viewed as connections between the corresponding points. Since the cones around these points remain unchanged by the above simplification, the instance with simplified Zone B is equivalent to the original with respect to $FO_{top}(\mathbb{P}, <_x, <_y)$ by [KPV97].

The constructions of *nice*(*cones*(I)) and *nice*(*cones*(J)) outlined so far have been independent of each other. However, we glossed over two technical subtleties:

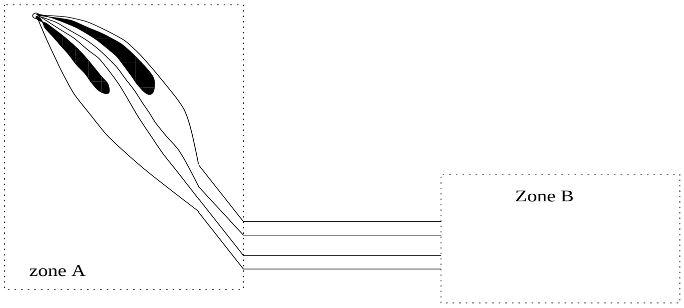

Figure 6: $nice(cones(I))$

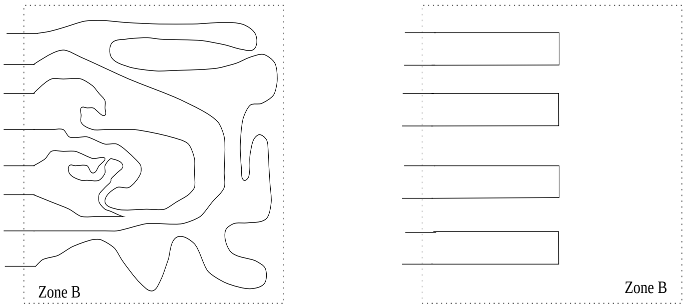

Figure 7: Zone B of *nice*

(i) It may be necessary to perform a reflection on one of the instances in order to make them equivalent with respect to the $FO^r(\mathbb{P}, <_x, <_y)$-game.

(ii) The construction of the instances in the first stage must make a choice of a first line or petal (the closest to the positive x-axis).

The information needed to deal with (i) and (ii) is provided by the winning strategy of the Duplicator in the FO^{r+2}-game on $cycles(I)$ and $cycles(J)$. Consider (i). To see if a reflection must be applied to one of the instances, consider the winning strategy of the Duplicator for the game where the Spoiler first picks an orientation in the structure $cycles(I)$. Duplicator responds by picking an orientation in $cycles(J)$. If the orientations are the same, no reflection is needed. Otherwise, a reflection is applied to one of the instances.

Now consider (ii). Continuing with the game, suppose Spoiler picks a node n_I in $cycles(I)$ and Duplicator responds by picking a node n_J in $cycles(J)$. Then in the construction of $nice(cones(I))$ the first line or petal is chosen to be the one represented in the invariant by n_I, and the first line or petal in $nice(cones(J))$ is chosen to be the one corresponding to n_J. The construction of $nice(cones(I))$ and $nice(cones(J))$ is now complete.

We next show that $nice(cones(I))$ and $nice(cones(J))$ are equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$, i.e. the Duplicator has a winning strategy for the $FO^r(\mathbb{P}, <_x, <_y)$-game on the two instances. This is done by combining winning strategies from two games. The first is used to guide the game within Zone A, and the second provides a strategy for Zone B. It is easy to see that the game in the two zones are independent of each other.

We begin by describing Duplicator's winning strategy for Zone A. The strategy mimics Duplicator's winning strategy for the r remaining moves of the $(r+2)$-game on $cycles(I)$ and $cycles(J)$ (recall that the first two moves are used in the construction of $nice(cones(I))$ and $nice(cones(J))$). Suppose Spoiler picks a pebble p in Zone A in, say, instance $nice(cones(I))$. If p is in the region, it belongs to an edge or face represented by a node c in $cycles(I)$. Suppose Duplicator's response in the game on $cycles(I)$ and $cycles(J)$ is to pick a node c' in $cycles(J)$. In the $FO^r(\mathbb{P}, <_x, <_y)$-game on $nice(cones(I))$ and $nice(cones(J))$ Duplicator picks a point p' on the edge or face corresponding to c'. Furthermore, due to the similar shapes of the two cones, p' can be picked so that it sits in the desired order relative to previously picked points. If the point

p lies outside the region, it can be associated with the point in the region immediately above it, and the strategy is similar.

Duplicator's winning strategy for Zone B is much simpler. Essentially, Zone B can be viewed as a total order (whose elements correspond to the pairs of connected lines). It is well-known that total orders are equivalent with respect to the FO^r-game iff they have at least 2^{r-1} elements (e.g., see [Ros82]). From the existence of the winning strategy for the FO^{r+2}-game on $cycles(I)$ and $cycles(J)$ it follows immediately that both structures have exactly the same number of edges (in this case, the strategy for the $FO^r(\mathbb{P}, <_x, <_y)$-game on Zone B is the identity) or at least 2^r edges, yielding total orders of size 2^{r-1} (pairs of connected lines). The winning strategy for the FO^r-game on total orders of size at least 2^{r-1} can be easily mimicked in the $FO^r(\mathbb{P}, <_x, <_y)$-game on Zone B of $nice(cones(I))$ and $nice(cones(J))$.

Lastly, the game in the intermediate zone between Zones A and B is very similar to the game in Zone B.

In conclusion, $nice(cones(I))$ and $nice(cones(J))$ are equivalent with respect to $FO^r(\mathbb{P}, <_x, <_y)$, and $cones(I)$ and $cones(J)$ are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$, which completes the proof. $\square$

We next consider instances with several cones. Let I and J be two spatial instances, and $cycles(I)$ and $cycles(J)$ be defined as earlier. Given $r > 0$, the r-type of a finite structure is the set of FO^r sentences it satisfies. Note that the FO^r sentences satisfied by each r-type can be used to order the r-types. Also recall that for each r there are finitely many r-types (the same as the number of equivalence classes of finite structures with respect to the FO^r-game). Let us now define the equivalence relation $\simeq^r$ on the structures $cycles(I)$ as follows: $cycles(I) \simeq^r cycles(J)$ iff for each $(r+2)$-type τ $cycles(I)$ and $cycles(J)$ contain the same number of single cycles of $(r+2)$-type τ, or both contain more than 2^r single cycles of type τ.

We can now extend Lemma 4.6 to the case of multiple cycles as follows:

Lemma 4.7 *Let I and J be spatial instances. If $cycles(I) \simeq^r cycles(J)$ then I and J are equivalent with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$.*

Proof Recall that 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.

We assume given an arbitrary order on $(r+2)$-types. For each $(r+2)$-type τ let a_τ be a fixed $(r+2)$-type of a node of cycle of r-type τ.

Similarly to Lemma 4.6, we construct instances $nice(cones(I))$ and $nice(cones(J))$ that are equivalent to $cones(I)$ and $cones(J)$ with respect to $FO^r_{top}(\mathbb{P}, <_x, <_y)$ and will be shown equivalent to each other with respect to $FO^r(\mathbb{P}, <_x, <_y)$. Basically, the construction for multiple cycles replicates the construction for individual cycles (see Figure 8). In particular, Zones A and B are the same for each cycle. Additionally, all cones are lined up in the order of their r-types, and the first edge or face in a cycle of $(r+2)$-type τ is chosen so that its corresponding node in $cycles(I)$ (resp. $cycles(J)$) has the $(r+2)$-type a_τ previously fixed.

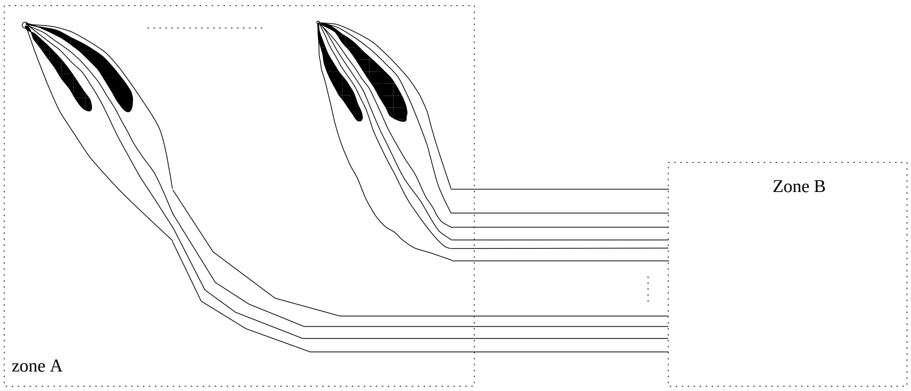

Figure 8: $nice(cones(I))$

The $FO^r(\mathbb{P}, <_x, <_y)$ game on $nice(cones(I))$ and $nice(cones(J))$ is now played as in Lemma 4.6, with the additional complication of having to choose a cone at each move. However, this is easy: if Spoiler plays in the k-th cone of type τ of which there are fewer than 2^r cones, Duplicator responds in the k-th cone of the same type τ (recall that by the definition of $\simeq^r$, $cycles(I)$ and $cycles(J)$ have in this case the same number of cycles of type τ). If there are more than 2^r cones of type τ, then Duplicator picks a cone according to the winning strategy for the FO^r game on orders of length 2^r (the order of cones of type τ with respect to $<_x$). With the cone properly selected, the winning strategy is the same as in the single cone case described in Lemma 4.6. $\square$

Lemma 4.7 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 of $\simeq^r$ on their cycles. Note that there are finitely many such equivalence classes and each can be defined by an FO_{inv} formula. For each equivalence class τ of $\simeq^r$ let ψ_τ be the FO_{inv} formula defining τ, i.e. $cycles(I) \models \psi_\tau$ iff $cycles(I)$ belongs to τ. If $I \models \varphi$ for each I such that $cycles(I) \in \tau$, we say that τ *satisfies* φ. Let $\mathcal{T}$ be the set of equivalence classes of $\simeq^r$ satisfying φ. The translation of φ into FO_{inv} is the sentence

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

(Recall from Lemma 4.5 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 shows that each $FO^r_{top}(\mathbb{P}, <_x, <_y)$ sentence φ has a corresponding sentence $inv(\varphi)$ in FO_{inv}. In order to *effectively* compute $inv(\varphi)$ one needs to compute the set $\mathcal{T}$ of equivalence classes of $\simeq^r$ satisfying φ. To this end, it suffices to do the following for each equivalence class τ:

(i) construct a cone instance I_τ such that $cycles(I_\tau)$ is in τ;

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

If the answer to (ii) is positive, then $\tau \in \mathcal{T}$. Clearly, verifying (ii) presents no problem once I_τ is constructed. The following lemma addresses (i), which is less simple.

Lemma 4.8 *Given a class τ of $\simeq^r$ one can effectively construct a cone instance I_τ such that $cycles(I_\tau)$ is in τ.*

Proof Recall that each equivalence class of $\simeq^r$ is characterized by the multiset of $(r+2)$-types of the individual cycles, with the multiplicity truncated to 2^r. Since a cone instance I_τ such that $cycles(I_\tau)$ is in τ is essentially a union of cone instances of single cycles of appropriate $(r+2)$-types, it is sufficient to show how to construct I_τ when τ is an $(r+2)$-type of a *single* cycle. Furthermore, because an $(r+2)$-type describes a cycle and its image obtained by reversing the orientation, we can assume that the orientation is fixed and construct an instance for the corresponding $(r+1)$-type.

Thus, let τ be an $(r+1)$-type of a single colored cycle. To construct a cycle of type τ (if such exists), it is useful to notice a strong connection between $(r+1)$-types of colored cycles and r-types of words over the alphabet consisting of the two colors. Indeed, following the choice of the first nodes in an $(r+1)$-game on two colored cycles, the remainder r-game is essentially played on the words obtained by going around the cycles starting from the nodes picked in the first move. Given a colored cycle with an orientation, let us say that a word on the alphabet of colors is *compatible* with the cycle if it is obtained by walking around the cycle in the given orientation, starting from some node. For a cycle c, let $L(c)$ be the set of words compatible with c, and for a class of cycles τ, let $L(\tau) = \bigcup_{c \in \tau} L(c)$. Clearly, there exists a cycle of type τ iff $L(\tau) \neq \emptyset$. We will show that $L(\tau)$ is regular and provide a way to compute a regular expression for it. Then it is possible to test non-emptiness of $L(\tau)$, and if $L(\tau) \neq \emptyset$ produce a word in $L(\tau)$ (which yields a cycle of type τ).

Note that an $(r+1)$-type of colored cycles determines the set of r-types of words compatible with such cycles. It well known that there are finitely many r-types of words over a fixed alphabet, and the r-equivalence classes are precisely the regular star-free languages of dot-depth r [RS97]. Furthermore, the number of such languages is hyperexponential in r and the regular expressions defining the languages of dot-depth r can be effectively computed [Wil]. Let L_μ denote the regular language consisting of words of r-type μ. Also, let R consist of the set of r-types μ such that $L_\mu \cap L(\tau) \neq \emptyset$ and $\bar{R}$ consist of the remaining r-types. Note that R (and so $\bar{R}$) is effectively computable from τ [EF95b]. Furthermore, it turns out that $L(\tau)$ can be computed from R. To see this, let us denote by $\tilde{L}$ the closure of a language L under *conjugate*, i.e. $\tilde{L} = \{vu \mid uv \in L\}$. (It is easily seen that if L is regular then $\tilde{L}$ is also regular.) We claim that

$$(\dagger) \quad L(\tau) = \bigcap_{\mu \in R} \tilde{L}_\mu \; - \; \bigcup_{\mu \in \bar{R}} \tilde{L}_\mu$$

Let L_R denote the right hand side of ($\dagger$), and observe that L_R is a regular language whose regular expression is effectively computable from R. To prove ($\dagger$), consider first the inclusion $L(\tau) \subseteq L_R$. Suppose c is of $(r+1)$-type τ. Clearly, $L(c) \cap L_\mu \neq \emptyset$ for each $\mu \in R$. Since $L(c)$ is generated by the conjugates of any word it contains, it follows that $L(c) \subseteq \tilde{L}_\mu$ for every $\mu \in R$. For similar reasons $L(c) \cap \tilde{L}_\mu = \emptyset$ for every $\mu \in \bar{R}$. It follows that $L(\tau) \subseteq L_R$.

Conversely, consider $L_R \subseteq L(\tau)$. Let $w \in L_R$ and c_w be the cycle corresponding to w. We wish to show that c_w has $(r+1)$-type τ. Let c be a cycle of $(r+1)$-type τ. It is sufficient to show that there is a winning strategy for the $(r+1)$-game on c_w and c. Suppose Spoiler begins by picking a node in c, yielding a corresponding word u of r-type μ. Clearly, $\mu \in R$ and since $w \in \bigcap_{\mu \in R} \tilde{L}_\mu$, some conjugate w' of w has r-type μ. Duplicator then picks the node in c_w yielding w'. In the remainder r moves, Duplicator mimics the winning strategy of the r-game on w' and u. Now suppose Spoiler's first move is to pick a node in c_w, yielding a conjugate w'' of w. Since $w \notin \bigcup_{\mu \in \bar{R}} \tilde{L}_\mu$ it follows that w'' has r-type ν for some $\nu \in R$. Since c has type τ, there must exist a node yielding a word u of type ν, which Duplicator picks. Again, the remainder of the game mimics the winning strategy in the r-game on w'' and u. Thus, c has $(r+1)$-type τ and so $w \in L(\tau)$, which completes the proof of ($\dagger$).

In summary, using ($\dagger$), one can compute as follows a cycle of type τ, if such exists:

- compute a regular expression for L_R;
- if $L_R = \emptyset$ then there is no cycle of type τ;
- if $L_R \neq \emptyset$ find a word $w \in L_R$ and build the corresponding cycle.

Finally, it is straightforward to construct from the cycle a corresponding cone instance I_τ.

Note that the complexity of the above procedure is hyperexponential in r because there are hyperexponentially many r-types of words, and therefore the size of the regular expression for L_R is hyperexponential in r. □

The above development leads to the main result on the translation.

Theorem 4.9 *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 from φ.*

Proof The FO_{inv} sentence $inv(\varphi)$ is the sentence described in ($\star$) above. This is first-order by Lemma 4.7. The computability follows from Lemma 4.8. □

Remarks (i) Recall that the topological invariant, as defined in the present paper, contains the relation *Between* providing the full cyclic order on the elements adjacent to a given vertex. Other presentations of the invariant (including the one in [PSV99]) only provides the *successor* on the element adjacent to each vertex. The distinction is relevant to the results in this section. (They do not affect the results of Section 3, since *fixpoint* can compute the cyclic order from the successor.) Indeed, the availability of the cyclic order is used in a crucial way in the proof of Theorem 4.9. Specifically, it allows to define $cycles(I)$ from $top(I)$ by first-order means. It turns out that Theorem 4.9 is no longer true if only the successor is provided. To see this, consider instances having only a single cone. Consider the two sets of such instances illustrated in Figure 9 : on the left side the cone consists of a face followed by several lines then two faces followed by a large number of lines, then again a face followed by lines, and finally two faces followed by lines; on the right side the single faces are consecutive, as are the two pairs of faces, again with a large number of lines in between. It is possible to find a $FO_{top}(\mathbb{R},<)$ sentence distinguishing the two sets. However, their topological invariants (where successor rather than cyclic order is provided) are indistinguishable by FO_{inv} sentences. This follows from the fact that for each $d > 0$ if there are sufficiently many lines the two instances have the same neighborhood types of size d. Then it follows from [FSV95] that the two sets cannot be distinguished by an FO_{inv} sentence. Thus, not all $FO_{top}(\mathbb{R},<)$ sentences can be translated into FO_{inv} sentences on the invariant if the cyclic order is not provided.

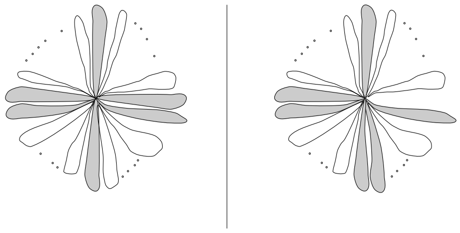

Figure 9: Instances that are distinguishable by $FO_{top}(\mathbb{R},<)$, but not by FO_{inv}

(ii) Theorems 4.2 and 4.9 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 hyperexponential to linear time if the target language is the more powerful $FO + \mu^+$ instead of FO_{inv}.

(iii) We have shown that all $FO_{top}(\mathbb{R},<)$ sentences 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 10. Clearly, FO_{inv} can distinguish between the two instances; it follows from [KPV97] that $FO_{top}(\mathbb{R},<)$ cannot.

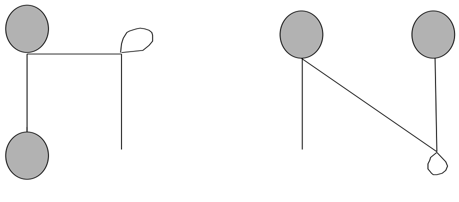

Figure 10: Instances I and J are distinguishable by FO_{inv} but not by $FO_{top}(\mathbb{R},<)$

(iv) In considering the translation issue, we focused on the topological fragment of $FO(\mathbb{R},<)$ on the spatial database side. It is natural to wonder whether the results can be extended to more powerful languages, such as $FO(\mathbb{R},<,+,*)$. However, there seem to be some difficulties involved. For the language $FO(\mathbb{R},<,+,*)$, the use of Ehrenfeucht-Fraissé games becomes extremely complicated, and it is not clear how to find the *nice* instances topologically equivalent to the input that we used in our proof to facilitate Duplicator's strategy. Incidentally, note that it is open whether the topological fragments of $FO(\mathbb{R},<,+,*)$ and $FO(\mathbb{R},<)$ coincide. This question is similar to the collapse of *order-generic* queries for finite database (see [BDLW98]).

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 hyperexponential 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, although (ii) is likely to be prohibitively expensive due to the hyperexponential complexity in the quantifier depth. 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*[8] 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. We plan to evaluate option (iv) experimentally in future work.

[8]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. Unfortunately, this cannot be extended to the general case of multiple regions [GroSeg99]. The complexity of the translation into FO_{inv} is prohibitively high, even for one-region schemas. However, it becomes more reasonable in several special cases of practical interest, such as the existence of a small bound on number of lines intersecting at one point. In all cases, the cost of the translation has to be balanced against the potential savings due to the difference in size between spatial databases and their topological invariants. 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 semi-algebraic 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.

[AV95] S. Abiteboul and V. Vianu. Computing with first-order logic. *Journal of Computer and System Sciences* 50(2):309–335, 1995.

[BCR98] J. Bochnak, M. Coste, M.-F. Roy. *Real Algebraic Geometry.* Springer Verlag, 1998.

[BDLW98] M. Benedikt, G. Dong, L. Libkin and L. Wong. Relational expressive power of constraint query languages. In Journal of the ACM, 45(1):1–34, 1998.

[BKR86] M. Ben-Or,D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. In *Journal of Computer and System Sciences*, 32(2):251–264, 1986.

[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.

[Cos89] M. Coste. Effective Semialgebraic Geometry. in Geometry and Robotics, LNCS 391, Springer-Verlag, 1989, pp 1-27.

[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(2):160–175, 1995.

[Ege93] M. Egenhofer. A Model for Detailed Binary Topological Relationships. *Geomatica*, 47(3-4): 261-273, 1993.

[EF91] M.J. Egenhofer and R. Franzosa. Point-set topological spatial relations. *Int. J. Geo. Info. Sys.*, 5(2):161–174, 1991.

[EF95a] M. Egenhofer and R. Franzosa. On the Equivalence of Topological Relations. *Int'l J. of Geographical Information Systems*, 9(2):133-152, 1995.

[EF95b] H-D Ebbinghaus and J. Flum. Finite Model Theory. Springer Verlag, 1995.

[FE92] R. Franzosa and M. Egenhofer. Topological Spatial Relations Based on Components and Dimensions of Set Intersections. In *Proc. SPIE's OE/Technology '92-Vision Geometry*, R. Melter and A. Wu, eds., Boston, MA, 1992, pp. 236-246.

[Ege94] M. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86-95, 1994.

[FK86] A. Frank and W. Kuhn. Cell Graph: A Provable Correct Method for the Storage of Geometry. In *Proc. Second International Symposium on Spatial Data Handling*, D. Marble, ed., Seattle, WA, 1986, pp. 411-436.

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

[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.

[GPP95] M. Grigni, D. Papadias and C.H. Papadimitriou. Topological Inference. *IJCAI*, 1995.

[Gro97] M. Grohe. Large Finite Structures with Few L^k-Types. In *Proc. Symp. on Logic in Computer Science*, 216–227, 1997.

[Gro98] M. Grohe. Fixed-point logics on planar graphs. In *Proc. Symp. on Logic in Computer Science*, 6–15, 1998.

[GroSeg99] M. Grohe and L. Segoufin. Personal communication, 1999.

[GK97] S. Grumbach and G. Kuper. Tractable recursion over geometric data. In *Proc. Int'l. Conf. on Constraint Programming*, 450–462, 1997.

[GS97] S. Grumbach and J. Su. Queries with Arithmetical Constraints. *Theoretical Computer Science*, 173(1):151-181, 1997.

[GS99] S. Grumbach and J. Su. Finitely representable databases. *Journal of Computer and System Sciences*, 55(2):273–298, 1997.

[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. Ricther et al., editor, *Computation and Proof Theory, Lecture Notes in Mathematics 1104*, 175–216. Springer-Verlag, 1984.

[Gur88] Y. Gurevich, *Logic and the challenge of computer science*, Trends in Theoretical Computer Science (E. Borger, ed.), Computer Science Press, 1–57, 1988.

[Her91] J. Herring. TIGRIS: A Data Model for an Object-Oriented Geographic Information System. *Computers and Geosciences*, 18(4):443–452, 1991.

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

[Imm86] N. Immerman. Relational queries computable in polynomial time. In *Information and Control*, 68(1-3):86–104, 1986.

[Imm87] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proc. 2nd Structure in Complexity Conference*, 194–202, 1987.

[KKR95] P.C. Kanellakis, G.M. Kuper and P.Z. Revesz. Constraint query languages. In *Journal of Computer and System Sciences*, 51(1):26–52, 1995.

[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*, 1–13, 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, 432–446.

[Mor85] S. Morehouse. ARC/INFO: A Geo-Relational Model for Spatial Information. In *Proc. Int'l. Symp. on Computer Assisted Cartography (Auto-Carto 7)*, 388-397, 1985.

[Mor89] S. Morehouse. The Architecture of ARC/INFO. In *Proc. Int'l. Symp. on Computer Assisted Cartography (Auto-Carto 9)*, 266-277, 1989.

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

[OV91] P. van Oosterom and T. Vijlbrief. Building a GIS on Top of the Open DBMS Postgres. *EGIS '91*, Brussels, Belgium, 1991, 775-787.

[PSV99] C.H. Papadimitriou, D. Suciu and V. Vianu. Topological Queries in Spatial Databases. In Journal of Computer and System Sciences 58(1): 29-53, 1999.

[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. IEEE Conf. on Logic in Computer Science*, 79–87, 1995.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. in *J. of Symbolic Computation, 13(3)*, 1992, pp. 255-300.

[Ros82] J. G. Rosenstein. *Linear Orderings.* Academic Press, 1982.

[RS97] G. Rozenberg and A. Salomaa. Handbook of formal languages. Volume 3, chapter 7, by Wolfgang Thomas. Springer, 1997.

[Seg] L. Segoufin. Manipulation de données spatiales et topologiques. PhD Thesis, Université d'Orsay, january 1999.

[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*, 2–11, 1993 .

[SZ91] P. Svensson and H. Zhexue. Geo-SAL: A Query Language for Spatial Data Analysis. In *Proc. Advances in Spatial Databases-Second Symposium*, SSD '91, O. Günther and H.-J. Schek, eds., Lecture Notes in Computer Science, vol. 525, Springer-Verlag, New York, NY, 1991, pp. 119-140.

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

[Wil] Thomas Wilke. Personal communication, 1998.

[Var82] M.Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, 1982, 137-146.

[Wor92] M. Worboys. A Geometric Model for Planar Geographical Objects. *Int'l. J. of Geographical Information Systems*, 6(5):353–372, 1992.