Expressing topological connectivity of spatial databases

Floris Geerts and Bart Kuijpers*

University of Limburg (LUC)
Department WNI
B-3590 Diepenbeek, Belgium
{floris.geerts, bart.kuijpers}@luc.ac.be

Abstract. We consider two-dimensional spatial databases defined in terms of polynomial inequalities and focus on the potential of programming languages for such databases to express queries related to *topological connectivity*. It is known that the topological connectivity test is *not* first-order expressible. One approach to obtain a language in which connectivity queries can be expressed would be to extend FO+Poly with a generalized (or Lindström) quantifier expressing that two points belong to the same connected component of a given database. For the expression of topological connectivity, extensions of first-order languages with recursion have been studied (in analogy with the classical relational model). Two such languages are *spatial Datalog* and FO+Poly+While. Although both languages allow the expression of non-terminating programs, their (proven for FO+Poly+While and conjectured for spatial Datalog) computational completeness makes them interesting objects of study.

Previously, spatial Datalog programs have been studied for more restrictive forms of connectivity (e.g., piece-wise linear connectivity) and these programs were proved to correctly test connectivity on restricted classes of spatial databases (e.g., linear databases) only.

In this paper, we present a spatial Datalog program that correctly tests topological connectivity of arbitrary compact (i.e., closed and bounded) spatial databases. In particular, it is guaranteed to terminate on this class of databases. This program is based on a first-order description of a known topological property of spatial databases, namely that locally they are conical.

We also give a very natural implementation of topological connectivity in FO+Poly+While, that is based on a first-order implementation of the *curve selection lemma*, and that works correctly on arbitrary spatial databases inputs. Finally, we raise the question whether topological connectivity of arbitrary spatial databases can also be expressed in spatial Datalog.

* Research done while this author was at the Department of Mathematics and Computer Science of the University of Antwerp (UIA) as post-doctoral research fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

1 Introduction

The framework of *constraint databases*, introduced by Kanellakis, Kuper and Revesz [10] (an overview of the area of constraint databases can be found in [14]), provides a rather general model for spatial databases [16]. In this context, a spatial database, which conceptually can be viewed as an infinite set of points in the real space, is finitely represented as a union of systems of polynomial equations and inequalities (in mathematical terminology, such figures are called *semi-algebraic sets* [3]). The set of points in the real plane that are situated between two touching circles together with a segment of a parabola, depicted in Figure 1, is an example of such a spatial database and it could be represented by the polynomial constraint formula

$$(x^2 + (y-1)^2 \leq 1 \wedge 25x^2 + (5y-4)^2 > 16) \vee (y^2 - x = 0 \wedge (0 \leq y \wedge x \leq 1)).$$

In this paper, we will restrict our attention to two-dimensional spatial databases, a class of figures that supports important spatial database applications such as geographic information systems (GIS).

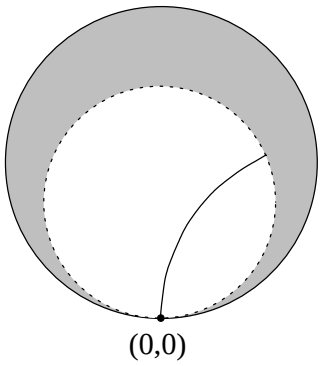

Fig. 1. An example of a spatial database.

In the past ten years, several languages to query spatial databases have been proposed and studied. A very natural query language, commonly known as FO+Poly, is obtained by extending the relational calculus with polynomial inequalities [16]. The query that returns the topological interior of a database S is expressed by the FO+Poly-formula

$$(\exists \varepsilon > 0)(\forall x')(\forall y')((x-x')^2 + (y-y')^2 < \varepsilon^2 \rightarrow S(x', y')),$$

with free variables x and y that represent the co-ordinates of the points in the result of the query. Although variables in such expressions range over the real numbers, FO+Poly queries can still be effectively computed [5, 18].

A combination of results by Benedikt, Dong, Libkin and Wong [2] and results of Grumbach and Su [6] implies that one cannot express in FO+Poly that a database is *topologically connected*. The topological connectivity test and the computation of connected components of databases are decidable queries [8, 17] and are of great importance in many spatial database applications, however.

One approach to obtain a language in which connectivity queries can be expressed would be to extend FO+Poly with a generalized (or Lindström) quantifier expressing that two points belong to the same connected component of a given database. In analogy with the classical graph connectivity query, which cannot be expressed in the standard relational calculus but which can be expressed in languages that typically contain a recursion mechanism (such as Datalog), we study extensions of FO+Poly with recursion for expressing topological connectivity, however. Two such languages are *spatial Datalog* and FO+Poly+While.

Both languages suffer from the well-known defect that their recursion, that involves arithmetic over an unbounded domain (namely polynomial inequalities over the real numbers), is no longer guaranteed to terminate. Therefore, these languages are *not* closed. FO+Poly+While is known to be a computationally complete language for spatial databases [7], however. Spatial Datalog is believed to be complete too [11, 13]. It is therefore interesting to establish the termination of particular programs in these languages (even be it by ad hoc arguments) as it is interesting to do this for programs in computationally complete general-purpose programming languages.

Spatial Datalog [10, 11, 13] essentially is Datalog augmented with polynomial inequalities in the bodies of rules. Programs written in spatial Datalog are not guaranteed to terminate. It is known that useful restrictions on the databases under consideration or on the syntax of allowed spatial Datalog programs are unlikely to exist [11]. As a consequence, termination of particular spatial recursive queries has to be established by ad-hoc arguments. On the other hand, if a spatial Datalog program terminates, a finite representation of its output can be effectively computed.

A first attempt [11] to express the topological connectivity test in this language consisted in computing a relation *Path* which contains all pairs of points of the spatial database which can be connected by a straight line segment that is completely contained in the database and by then computing the transitive closure of the relation *Path* and testing whether the result contains all pairs of points of the input database. In fact, this program tests for *piece-wise linear connectivity*, which is a stronger condition than connectivity. The program, however, cannot be guaranteed to work correctly on non-linear databases [11]: it experiences both problems with termination and with the correctness of testing connectivity (as an illustration: the origin of the database of Figure 1 (a) is a cusp point and cannot be connected to any interior point of the database by means of a finite number of straight line segments).

In this paper, we follow a different approach that will lead to a correct implementation of topological connectivity queries in spatial Datalog for compact

database inputs. In our approach we make use of the fact that locally around each of its points a spatial database is conical [3]. Our implementation first determines (in FO+Poly) for each point a radius within which the database is conical. Then all pairs of points within that radius are added to the relation *Path* and, finally we use the recursion of spatial Datalog to compute the transitive closure of the relation *Path*.[1]

We raise the question whether topological connectivity of arbitrary (not necessarily compact) spatial databases can be implemented in spatial Datalog. It can be implemented in FO+Poly+While, the extension of FO+Poly with a while-loop. FO+Poly+While is a computationally complete language for spatial databases [7], and therefore the known algorithms to test connectivity (One of the oldest methods uses homotopy groups computed from a CAD [17]. A more recent and more efficient method uses Morse functions [9]) can be implemented in this language. Our implementation is a very natural one, however. It is based on a constructive version of the *curve selection lemma* of semi-algebraic sets [3, Theorem 2.5.5]. We show that this curve selection can be performed in FO+Poly. Also in this implementation the transitive closure of a relation *Path* (this time initialized using an iteration) is computed. Once this transitive closure is computed, a number of connectivity queries, such as "Is the spatial database connected?", "Return the connected component of the point p in the database", "Are the points p and q in the same connected component of the database?" can be formulated. Grumbach and Su give examples of other interesting queries that can be reduced to connectivity [6].

Both of the spatial Datalog and of the FO+Poly+While implementation we prove they are guaranteed to terminate and to give correct results.

This paper is organized as follows. In the next section we define spatial databases and the mentioned query languages and recall the property that spatial databases are locally conical. In Section 3, we will describe our spatial Datalog and FO+Poly+While implementations. In Section 4, we will prove their correctness and termination.

2 Preliminaries

In this section, we define spatial databases and three query languages for spatial databases. Let $\mathbf{R}$ denote the set of the real numbers, and $\mathbf{R}^2$ the real plane.

2.1 Definitions

Definition 1. A *spatial database* is a geometrical figure in $\mathbf{R}^2$ that can be defined as a Boolean combination (union, intersection and complement) of sets of the form $\{(x, y) \mid p(x, y) > 0\}$, where $p(x, y)$ is a polynomial with integer coefficients in the real variables x and y.

[1] In fact, for our purposes it would suffice to consider the extension of FO+Poly with an operator for transitive closure, rather than the full recursive power of spatial Datalog.

Note that $p(x, y) = 0$ is used to abbreviate $\neg(p(x, y) > 0) \wedge \neg(-p(x, y) > 0)$.

In this paper, we will use the relational calculus augmented with polynomial inequalities, FO+Poly for short, as a query language.

Definition 2. A formula in FO+Poly is a first-order logic formula built using the logical connectives and quantifiers from two kinds of atomic formula: $S(x, y)$ and $p(x_1, \ldots, x_k) > 0$, where S is a binary relation name representing the spatial database and $p(x_1, \ldots, x_k)$ is a polynomial in the variables $x_1, \ldots, x_k$ with integer coefficients.

Variables in such formulas are assumed to range over $\mathbf{R}$. A second query language we will use is FO+Poly+While.

Definition 3. A program in FO+Poly+While is a finite sequence of *statements* and *while-loops*. Each statement has the form $R := \{(x_1, \ldots, x_k) \mid \varphi(x_1, \ldots, x_k)\}$, where φ is an FO+Poly formula that uses the binary relation name S (of the input database) and previously introduced relation names. Each while-loop has the form **while** φ **do** P **od**, where P is a program and φ an FO+Poly formula that uses the binary relation name S and previously introduced relation names.

The semantics of a program applied to a spatial databases is the operational, step by step execution. Over the real numbers it is true that for every computable constraint query, such as connectivity, there is an equivalent FO+Poly+While program.

A restricted class of FO+Poly+While programs consists of programs in spatial Datalog.

Definition 4. *Spatial Datalog* is Datalog where,

1. The underlying domain is $\mathbf{R}$;
2. The only EDB predicate is S, which is interpreted as the set of points in the spatial database (or equivalently, as a binary relation);
3. Relations can be infinite;
4. Polynomial inequalities are allowed in rule bodies.

We interpret these programs under the the bottom-up semantics. To conclude this section, we remark that a well-known argument can be used to show that FO+Poly can be expressed in (recursion-free) spatial Datalog with stratified negation [1]. In this paper we also admit stratified negation in our Datalog program.

2.2 Spatial databases are locally conical

Property 1 ([3], Theorem 9.3.5). For a spatial database A and a point p in the plane there exists a radius ε_p such that for each $0 < \varepsilon < \varepsilon_p$ holds that $B^2(p, \varepsilon) \cap A$ is isotopic to the cone with top p and base $S^1(p, \varepsilon) \cap A$.[2]

[2] With $B^2(p, \varepsilon)$ we denote the closed disk with center p and radius ε and with $S^1(p, \varepsilon)$ its bordering circle. A homeomorphism $h : \mathbf{R}^2 \to \mathbf{R}^2$ is continuous bijective function

We remark that a spatial database is also conical towards infinity. In the next section, we will show that such a radius ε_p, can be uniformly defined in FO+Poly.

The database of Figure 1 is locally around the origin isotopic to the cone that is shown in Figure 2. It is a basic property of semi-algebraic sets that the base $S^1(p,\varepsilon) \cap A$ is the finite union of points and open arc segments on $S^1(p,\varepsilon)$ [3]. We will refer to the parts of $B^2(p,\varepsilon) \cap A$ defined by these open intervals and points as the *sectors of p in A*. In the example of Figure 2, we see that the origin has five sectors: two arc segments of the larger circle, a segment of the parabolic curve and two areas between the two circles. Sectors are *curves* or *fully two-dimensional*.

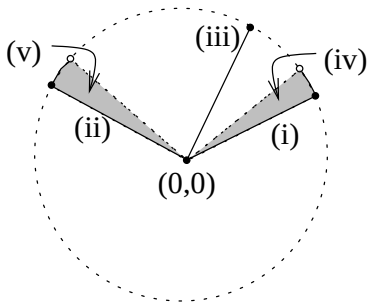

Fig. 2. The cone of $(0,0)$ of the spatial database in Figure 1.

In the next sections, we will use the following property. It can be proven similarly as was done for closed spatial databases [12].

Property 2. Let A be a spatial database. Then the following holds:

1. Only a finite number of cone types appear in A;
2. A can only have infinitely many points of five cone types (interior points, points on a smooth border of the interior that (don't) belong to the database, points on a curve, points on a curve of the complement);
3. The number of cone types appearing in A is finite and hence the number of points in A with a cone different from these five is finite.

The points with a cone of the five types mentioned in (2) of Property 2 are called *regular* points of the database. Non-regular points are called *singular*. We remark that the regularity of a point is expressible in FO+Poly [12].

whose inverse is also continuous. An isotopy of the plane is an orientation-preserving homeomorphism. Two sets are said to be *isotopic* if there is an isotopy that maps one to the other.

3 Connectivity queries in spatial Datalog

In general, a set S of points in the plane is defined to be *topologically connected* if it cannot be partitioned by two disjoint open subsets. This second-order definition seems to be unsuitable for implementation in spatial Datalog. Fortunately, for spatial databases S, we have the property that S is topologically connected if and only if S is *path connected* [3, Section 2.4] (i.e., if and only if any pair of points of S can be connected by a semi-algebraic curve that is entirely contained in S). In this section, we will show that for compact spatial databases path connectivity *can* be implemented in spatial Datalog and that for arbitrary databases it can be implemented in FO+Poly+While.

3.1 A program in spatial Datalog with stratified negation for connectivity of compact spatial databases

The spatial Datalog program for testing connectivity that we describe in this section is given in Figure 3.

$$
\begin{aligned}
\mathit{Path}(x,y,x',y') &\longleftarrow \varphi_{\mathrm{cone}}(S,x,y,x',y')\\
\mathit{Obstructed}(x,y,x',y') &\longleftarrow \neg S(\bar{x},\bar{y}),\ \bar{x}=a_1t+b_1,\ \bar{y}=a_2t+b_2,\\
&\qquad 0\le t,t\le 1, b_1=x, b_2=y,\\
&\qquad a_1+b_1=x', a_2+b_2=y'\\
\mathit{Path}(x,y,x',y') &\longleftarrow \neg \mathit{Obstructed}(x,y,x',y')\\
\mathit{Path}(x,y,x',y') &\longleftarrow \mathit{Path}(x,y,x'',y''),\ \mathit{Path}(x'',y'',x',y')\\
\mathit{Disconnected} &\longleftarrow S(x,y),\ S(x',y'),\ \neg \mathit{Path}(x,y,x',y')\\
\mathit{Connected} &\longleftarrow \neg \mathit{Disconnected}.
\end{aligned}
$$

Fig. 3. A program in spatial Datalog with stratified negation for topological connectivity of compact databases.

The first rule is actually an abbreviation of a spatial Datalog program that computes an FO+Poly formula $\varphi_{\mathrm{cone}}(S,x,y,x',y')$ that adds to the relation *Path* all pairs of points $((x,y),(x',y')) \in S\times S$ such that (x',y') is within distance $\varepsilon_{(x,y)}$ of (x,y), where $\varepsilon_{(x,y)}$ is such that S is conical (in the sense of Property 1) in $B^2((x,y),\varepsilon_{(x,y)})$. We will make the description of $\varphi_{\mathrm{cone}}(S,x,y,x',y')$ more precise below. Then all pairs of points of the spatial database are added in the relation *Path* which can be connected by a straight line segment that is completely contained in the database. Next, the transitive closure of the relation *Path* is computed and in the final two rules of the program of Figure 3 it is tested whether the relation *Path* contains all pairs of points of the input database.

Variations of the last two rules in the program of Figure 3 can then be used to formulate several connectivity queries (e.g., the connectivity test or the computation of the connected component of a given point p in the input database).

The description of $\varphi_{\mathrm{cone}}(S,x,y,x',y')$ in FO+Poly will be clear from the proof of the following theorem.

Theorem 1. *There exists an* FO+Poly *formula that returns for a given spatial database A and a given point p a radius ε_p such that the database A is conical within $B^2(p, \varepsilon_p)$ (in the sense of Property 1).*

Proof. (Sketch) Let A be a spatial database and p be a point. If p is an interior point of A, this is trivial. Assume that p is not an interior point of A. We compute within the disk $B^2(p, 1)$ the set $\gamma_{A,p}$ in FO+Poly. For each $\varepsilon \leq 1$, $S^1(p, \varepsilon) \cap A$ is the disjoint union of a finite number of intervals and points. We then define $\gamma_{A,p} \cap S^1(p, \varepsilon)$ to consists of these points and the midpoints of these intervals. For the database A of Figure 4 (a), $\gamma_{A,p}$ is shown in (b) of that figure in full lines and $\gamma_{A^c,p}$ is shown in dotted lines. These sets can be defined in FO+Poly using the predicate $Between_{p,\varepsilon}(x', y', x_1, y_1, x_2, y_2)$. $Between_{p,\varepsilon}(x', y', x_1, y_1, x_2, y_2)$ expresses for points (x', y'), (x_1, y_1) and (x_2, y_2) on $S^1(p, \varepsilon)$ that (x', y') is equal to (x_1, y_1) or (x_2, y_2) or is located between the clockwise ordered pair of points $((x_1, y_1), (x_2, y_2))$ of $S^1(p, \varepsilon)$ (for a detailed description of the expression of this relation in FO+Poly we refer to [12]).

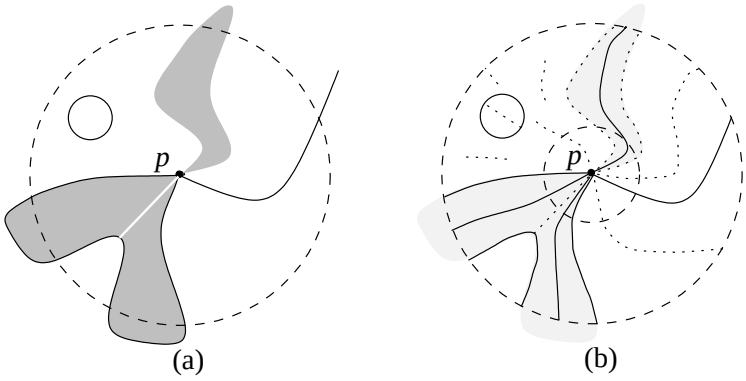

Fig. 4. The 1-environment of a database around p (a) and the construction to determine an ε_p-environment (smaller dashed circle) in which the database is conical (b). In (b), $\gamma_{A,p}$ is given in full lines and $\gamma_{A^c,p}$ in dotted lines.

Next, the singular points of $\gamma_{A,p} \cup \gamma_{A^c,p}$ can be found in FO+Poly. Let d be the minimal distance between p and these singular points. Any radius strictly smaller than d, e.g., $\varepsilon_p = d/2$, will satisfy the condition of the statement of this Theorem.

Then $B^2(p, \varepsilon_p) \cap (\gamma_{A,p} \cup \gamma_{A^c,p})$ consists of a finite number of non-intersecting (simple Jordan) curves starting in points $S^1(p, \varepsilon_p) \cap (\gamma_{A,p} \cup \gamma_{A^c,p})$ and ending in p and that for every $\varepsilon \leq \varepsilon_p$ each have a single intersection point with $S^1(p, \varepsilon)$. It is easy (but tedious) to show that there is an isotopy that brings $B^2(p, \varepsilon_p) \cap (\gamma_{A,p} \cup \gamma_{A^c,p})$ to the cone with top p and base $S^1(p, \varepsilon_p) \cap (\gamma_{A,p} \cup \gamma_{A^c,p})$. This is the isotopy we are looking for. □

3.2 An FO+Poly+While program for connectivity of arbitrary spatial databases

For compact spatial databases, all sectors of a boundary point p are all in the same connected component of the database (because a boundary point is always part of the database). Therefore all pairs of points in an ε_p-environment of p, can be added to the relation *Path*, even if they are in different sectors of p. For arbitrary databases, the sectors of a point $p \in \partial S \setminus S$[3] are *not necessarily* in the same connected component of the database.[4] This means that in general only pairs of points can be added to the relation *Path* if they are in the same sector of a point. We can achieve this by iteratively processing all sectors of the border points and adding only pairs of points that are in the same sector of a border point to the relation *Path*. For this iterative process we use the language FO+Poly+While. The resulting program is shown in Figure 5.

As in the compact case, we first initialize a relation *Path* and end with computing the transitive closure of this relation.

In the initialization part of the program, first all pairs of points which can be connected by a straight line segment lying entirely in the database, are added to the relation *Path*. Then, a 5-airy relation *Current* is maintained (actually destroyed) by an iteration that, as we will show, will terminate when the relation *Current* will have become empty. During each iteration step the relation *Path* is augmented with, for each border point p of the database, all pairs of points on the midcurve of the sector of p that is being processed during the current iteration.

The remainder of this section is devoted to the description of the implementations in FO+Poly of the algorithms *INIT*, *SeRA* (sector removal algorithm) and *CSA* (curve selection algorithm) of Figure 5. The correctness and termination of the resulting program will be proved in the next section.

The relation *Current* will at all times contain tuples $(x_p, y_p, \varepsilon, x, y)$ where (x_p, y_p) range over the the border points of the input database A, where ε is a radius and where (x, y) belong to a set containing the part of the ε-environment of (x_p, y_p) that still has to be processed. Initially, $INIT(S, x_p, y_p, \varepsilon_p, x, y)$ sets the relation *Current* to the set of five-tuples

$$\{(x_p, y_p, \varepsilon_p, x, y) \mid (x_p, y_p) \in \partial S, \varepsilon_p = 1, (x, y) \in B^2((x_p, y_p), \varepsilon_p) \cap S\}.$$

It is clear that *INIT* can be defined in FO+Poly.

Next, for all border points $p = (x_p, y_p)$ of the database, both in *SeRA* and *CSA* the "first sector" of p in the relation $Current(x_p, y_p, \varepsilon_p, x, y)$ will be determined. This is implemented as follows. We distinguish between a sector that is a curve and a fully two-dimensional one. We look at the latter case (the former is similar).

[3] We denote the topological border of S by ∂S.

[4] The same is true for the point at infinity, which can be considered as a boundary point of the database that does not belong to the database. To improve readability, we only consider *bounded* inputs in this section.

$Path := \{(x, y, x', y') \mid S(x, y) \wedge S(x', y') \wedge \overline{(x, y)(x', y')} \subseteq S\}$
$Current := INIT(S, x_p, y_p, \varepsilon, x, y)$
while $Current \neq \emptyset$ **do**
$\quad Current := SeRA(Current(x_p, y_p, \varepsilon, u, v), \varepsilon^{\text{new}}, x, y)$
$\quad Path := CSA(Current(x_p, y_p, \varepsilon, u, v), x, y, x', y')$
od
$Y := \emptyset$
while $Y \neq Path$ **do**
$\quad Y := Path;$
$\quad Path := Path \cup \{(x, y, x', y') \mid (\exists x'')(\exists y'')(Path(x, y, x'', y'') \wedge$
$\qquad\qquad Path(x'', y'', x', y'))\}$
od.

Fig. 5. An FO+Poly+While program for topological connectivity of arbitrary databases. The notation $\overline{(x, y)(x', y')}$ stands for the line segment between the points (x, y) and (x', y').

We define an order on the circle $S^1(p, \varepsilon)$ with $0 < \varepsilon < \varepsilon_p$, by using the relation $Between_{p,\varepsilon}(x', y', x_1, y_1, x_2, y_2)$, and by taking the point $p + (0, \varepsilon)$ as a starting point (see proof of Theorem 1). For each $0 < \varepsilon < \varepsilon_p$, the intersection of the "first fully two-dimensional sector" with $S^1(p, \varepsilon)$ is defined as the first (using the just defined order) open interval on this circle. This is clearly dependent on the radius ε. For the database of Figure 6 (a) this dependency is illustrated in (b) of that figure. The "first sector" falls apart into four parts (shaded dark), depending on the radius ε. Furthermore, as in Theorem 1, the first midcurve, i.e. the midcurve of the "first sector", within radius ε_p is computed in FO+Poly (the thick curve segments in Figure 6 (b)). By our definition of the "first sector", this first midcurve needs not to be connected. Hence, we obviously do not want to add *all* pairs (q, q') of points in this set to the relation *Path*.

We can, however, compute a new (and smaller) $\varepsilon_p^{\text{new}}$ such that the curve of midpoints has no longer singular points within $B^2(p, \varepsilon_p^{\text{new}})$. In Figure 6 (b), the small dashed circle around p has radius $\varepsilon_p^{\text{new}}$. Within the radius $\varepsilon_p^{\text{new}}$ the midcurve is connected and the point p belongs to its closure.

SeRA now updates ε_p in the relation *Current* to $\varepsilon_p^{\text{new}}$ and removes the first sector from the relation *Current*. This means that the set of points (x, y) that are in the relation *Current* with the point $p = (x_p, y_p)$ will initially be $B^2(p, \varepsilon_p) \cap A$, then $B^2(p, \varepsilon_p^{\text{new}}) \cap A$ minus the first sector of p (after the first iteration), then $B^2(p, \varepsilon_p^{\text{new}'}) \cap A$ minus the first two sectors of p (after the second iteration), etc.

CSA will add to the relation *Path*, all pairs (q, q') of midpoints at different distances $\varepsilon, \varepsilon' < \varepsilon_p^{\text{new}}$ from p (ε can be taken 0, if p belongs to the database) of the sector that has just been removed by *SeRA*.

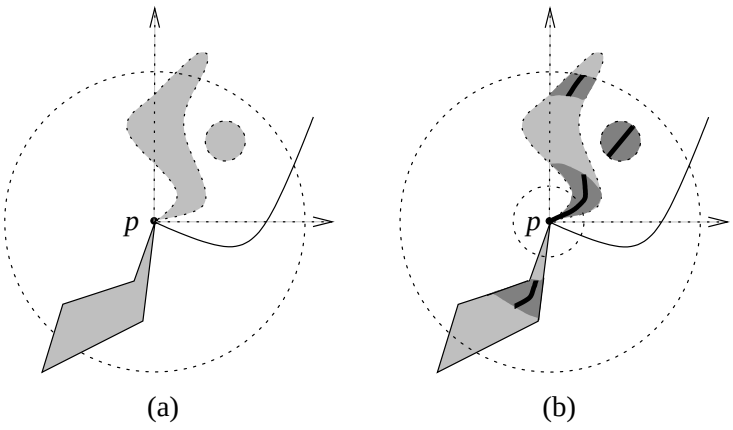

Fig. 6. The ε_p-environment of the point p in (a) and the "first sector" of p, the midcurve of the "first sector" and $\varepsilon_p^{\text{new}}$ in (b).

4 Correctness and termination of the programs

In this section, we prove the correctness and termination of both programs of the previous section.

Theorem 2. *The spatial Datalog program of the previous section correctly tests connectivity of compact spatial databases and the* FO+Poly+While *program of the previous section correctly tests connectivity of arbitrary spatial databases. In particular, the spatial Datalog program is guaranteed to terminate on compact input databases and* FO+Poly+While *program terminates on all input databases.*

Proof. (Sketch) To prove *correctness,* we first have to verify that for every input database S our programs are *sound* (i.e., two points in S are in the same connected component of S if and only if they are in the relation *Path*). Secondly, we have to determine the *termination* of our programs (i.e., we have to show that the first while-loop in Figure 5 that initializes the relation *Path* ends after a finite number of steps and that for both programs the computation of the transitive closure of the relation *Path* ends). To prove the latter it is sufficient that we show that there exists a bound $\alpha(S)$ such that any two points in the same connected component of S end up in the relation *Path* after at most $\alpha(S)$ iterations of the computation of the transitive closure. To improve readability, we only consider *bounded* inputs of the FO+Poly+While program in this proof.

Soundness. The if-implication of soundness (cf. supra) is trivial. So, we concentrate on the only-if implication. We use Collins's Cylindrical Algebraic Decomposition (CAD) [5] to establish the only-if direction. This decomposition returns for a polynomial constraint description of S, a decomposition $c(S)$ of

the plane in a finite number of cells. Each cell is either a *point*, a 1-dimensional *curve* (without endpoints), or a two-dimensional open *region*. Moreover, every cell is either part of S or of the complement of S. In order to prove that any two points in the same connected component of S are in the transitive closure of the relation *Path*, it is sufficient to prove this for

1. any two points of S that are in one cell of $c(S)$, in particular,
   a. two points of S that are in the same region,
   b. two points of S that are on the same curve, and
2. any two points of S that are in adjacent cells of $c(S)$, in particular,
   a. a point that is in a region and a point that is on an adjacent curve,
   b. a point that is in a region and an adjacent point,
   c. a point that is on a curve and an adjacent point.

1.a. In this case the two points p and q are part of a region in the interior of S, they can be connected by a semi-algebraic curve γ lying entirely in the interior of S [3]. Since uniformly continuous curves (such as semi-algebraic ones) can be arbitrarily closely approximated by a piece-wise linear curve with the same endpoints [15], p and q can be connected by a piece-wise linear curve lying entirely in the interior of S, we are done.
1.b. The curves in the decomposition are either part of the boundary of S or vertical lines belonging to S. In the latter case, the vertical line itself connects the two points. For the former case, let p and q be points on a curve γ in the cell decomposition. We prove for the case of the FO+Poly+While program that p and q are in the transitive closure of the relation *Path*. For the spatial Datalog program the proof is analogous. Let γ_{pq} be the curve segment of γ between p and q. Since all points r on γ_{pq} belong to the border of S, the algorithm *SeRA* processes the curve γ twice as sectors of r. We cover γ_{pq} with disks $B^2(r, \varepsilon_r)$, where ε_r is the radius constructed by *SeRA* when processing γ as a sector of r. Since γ_{pq} is a compact curve this covering has a finite sub-covering of, say, m closed balls. Then, the points p and q are in the relation *Path* after m iterations in the computation of the transitive closure.
2.a. A point on a vertical border line of a region can be connected by one single horizontal line with a point in the adjacent region. Hence, this case reduces to Case 1.a. If the point is on a non-vertical boundary curve of S, the sector around that point, intersecting the adjacent region contains a midcurve, connecting the point to the interior of the adjacent region (in the case of the spatial Datalog program even more pairs are added). Again this case reduces to Case 1.a.

2.b. In this case there is a midcurve from p in to the two-dimensional sector intersecting the region cell in $c(S)$. We distinguish between two cases. These two cases are depicted in Figure 7. In (a) the midcurve intersects the cell, while in (b) this is not the case. In Case (a), point p is connected by this midcurve to the cell, hence this case reduces to Case 1.a. For Case (b), we let r be a midpoint of a curve computed by *SeRA* belonging to the connected component of the interior of S that contains q. Hence, after using a similar argument as in Case 1.a, p and q belong to the transitive closure of the relation *Path* via a curve that passes

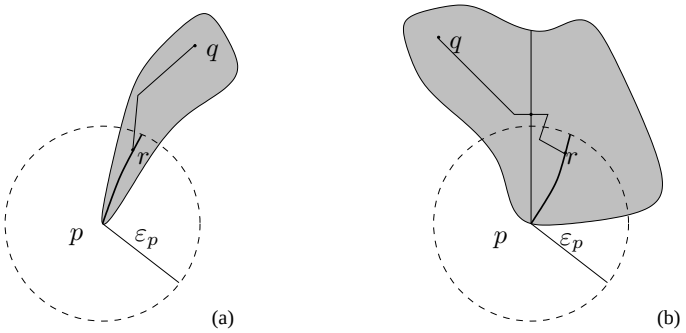

Fig. 7. The two cases in 2.b.

through r. the vertical line through p adjacent to the region.
2.c. There are various cases to be distinguished here. A vertical curve can be dealt with as before. A non-vertical curve is either a curve belonging to the border of S or a curve belonging to the interior of S. The latter case can be dealt with like in Case 2.b. For the former case, the algorithm *SeRA* will add p and a point of the border curve to the relation *Path*. The desired path to the border point can be found as in Case 1.b.

Termination. The first while-loop of the program in Figure 5 terminates since every border point of a spatial database has only a finite number of sectors in its cone and furthermore this number is bounded (this follows immediately from Property 2). After a finite number of runs of *SeRA*, the relation *Current* will therefore become empty.

To prove the termination of the computation of the transitive closure of the relation *Path* in both programs, we return to Collins's CAD. From the soundness-proof it is clear that it is sufficient to show that there exists an upper bound $\alpha(c)$ on the number of iterations of the transitive closure to connect two points in a cell c of $c(S)$.

We now show that for each region (i.e., two-dimensional cell) c in the CAD, there is a transversal $\gamma(c)$ in the relation *Path* from the bottom left corner of c to the upper right corner of c of finite length $\beta(c)$. Any two points of c can then be connected by at most $\alpha(c) = \beta(c)+2$ iterations of the transitive closure of the relation *Path* (namely by vertically connecting to the transversal and following it). For this we remark that the bottom left corner point of the cell c can be connected by a finite and fixed number of steps (see proof of soundness) with a point p in the interior of c. Similarly, the upper right corner point of c can be connected to some interior point q of c. The points p and q can be connected by a piece-wise linear curve, consisting of $\beta(c)$ line segments. This gives the desired transversal $\gamma(c)$. For cells c which are vertical line segments or single points the upper bound is 1. Remark that points on a one-dimensional cell c can also be connected by a finite number $\alpha(c)$ of line segments. This follows from the compactness of the curves (see Case 1.b of the soundness proof). □

5 Discussion

It is not clear whether the first while-loop of the FO+Poly+While program of Figure 5, which initializes the *Path* relation, can be expressed in spatial Datalog with stratified negation. More generally, we can wonder about the following.

Question: Can spatial Datalog with stratified negation express all computable spatial database queries?

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. *Journal of the ACM*, 45(1):1–34, 1998.
3. J. Bochnak, M. Coste, and M.-F. Roy. *Géométrie Algébrique Réelle*. Springer-Verlag, Berlin, 1987 (also *Real Algebraic Geometry*. Springer-Verlag, Berlin, 1998).
4. B.F. Caviness and J.R. Johnson (eds.) *Quantifier Elimination and Cylindrical Algebraic Decomposition* Springer-Verlag, Wien New York, 1998.
5. G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, Berlin, 1975. Springer-Verlag.
6. S. Grumbach and J. Su. Finitely representable databases. *Journal of Computer and System Sciences*, 55(2):273–298, 1997.
7. M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometrical query languages. in *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 62–67, ACM Press, New York, 1997.
8. J. Heintz, T. Reico, and M.-F. Roy. Algorithms in Real Algebraic Geometry and Applications to Computational Geometry. *Discrete and Computational Geometry: Selected Papers from the DIMACS Special Year*, Eds. J.E. Goodman, R. Pollack and W. Steiger, AMS and ACM, 6:137–164, 1991.
9. J. Heintz, M.-F. Roy, and P. Solernò. Description of the Connected Components of a Semi-Algebraic Set in Single Exponential Time. *Discrete and Computational Geometry*, 11: 121–140, 1994.
10. P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995 (Originally in *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pages 299–313, ACM Press, New York, 1990).
11. B. Kuijpers, J. Paredaens, M. Smits, and J. Van den Bussche. Termination properties of spatial Datalog programs. In D. Pedreschi and C. Zaniolo, editors, *Proceedings of "Logic in Databases"*, number 1154 in Lecture Notes in Computer Science, pages 101–116, Berlin, 1996. Springer-Verlag.
12. B. Kuijpers, J. Paredaens, and J. Van den Bussche. Topological elementary equivalence of closed semi-algebraic sets in the real plane. *The Journal of Symbolic Logic*, to appear, 1999.
13. B. Kuijpers and M. Smits. On expressing topological connectivity in spatial Datalog. In V. Gaede, A. Brodsky, O. Gunter, D. Srivastava, V. Vianu, and M. Wallace,

editors, *Proceedings of Workshop on Constraint Databases and their Applications*, number 1191 in Lecture Notes in Computer Science, pages 116–133, Berlin, 1997. Springer-Verlag.
14. G. Kuper, L. Libkin, and J. Paredaens. *Constraint databases*. Springer-Verlag, 2000.
15. E.E. Moise. *Geometric topology in dimensions 2 and 3*, volume 47 of *Graduate Texts in Mathematics*. Springer, 1977.
16. J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems*, pages 279–288, ACM Press, New York, 1994.
17. J.T. Schwartz and M. Sharir. On the piano movers' problem II. In J.T. Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 51–96. Ablex Publishing Corporation, Norwood, New Jersey, 1987.
18. A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 1951.