Termination properties of spatial Datalog programs

Bart Kuijpers, Jan Paredaens, Marc Smits and Jan Van den Bussche*

University of Antwerp**

Abstract. We consider spatial databases defined in terms of polynomial inequalities, and investigate the use of Datalog as a query language for such databases. Recursive programs are not guaranteed to terminate in this setting. Through a series of examples we show that useful restrictions on the databases under consideration or on the syntax of allowed programs, guaranteeing termination, are unlikely to exist. Hence, termination of particular recursive spatial queries must be established by ad-hoc arguments, if it can be established at all. As an illustration of the difficulties that can be encountered in this respect we discuss the topological connectivity query.

1 Introduction

The framework of *constraint databases*, introduced by Kanellakis, Kuper and Revesz [8], provides an elegant and powerful model of spatial databases [11]. In this setting, a spatial database, viewed as a possibly infinite set of points in real space, is represented as a Boolean combination of polynomial inequalities. For example, the unit disk in the real plane except its upper left quadrant would be represented as

$$\{(x, y) \mid x^2 + y^2 \leq 1 \wedge \neg(x \geq 0 \wedge y \geq 0)\}.$$

By extending the relational calculus with polynomial inequalities one obtains a simple spatial query language. For example, the query whether the database S contains a circle as a subset can be expressed as

$$(\exists x_0)(\exists y_0)(\exists r \neq 0)(\forall x)(\forall y)((x - x_0)^2 + (y - y_0)^2 = r^2 \Rightarrow S(x, y)).$$

Although variables now range over the whole of the real numbers, queries expressed in the calculus can still be effectively computed using methods from symbolic computation.

The expressiveness of the calculus is limited, however. For example, the combined results of Benedikt et al. [2] and Grumbach and Su [6] imply that one cannot express in the calculus that the database is topologically connected. The

* Post-doctoral research fellow of the Belgian National Fund for Scientific Research.

** Address: UIA, Informatica, Universiteitsplein 1, B-2610 Antwerpen, Belgium. Email: {kuijpers, pareda, msmits, vdbuss}@uia.ua.ac.be.

topological connectivity query can be viewed as the spatial analogue of the standard relational query of graph connectivity, which is also not expressible in the standard relational calculus. In order to be able to express queries such as graph connectivity, in the standard relational context, one typically uses a more powerful query language such as Datalog [17], an extension of the relational calculus with recursion.

It is therefore natural to likewise extend the calculus with recursion in the spatial context. However, we then face the well-known fact that recursion involving arithmetic over an unbounded domain, such as the polynomial inequalities over the reals in our setting, is no longer guaranteed to terminate. In other words, the property of effective computability of queries in the spatial calculus is lost when extending the calculus with recursion. For example, the following trivial program does not terminate:

$R(0) \longleftarrow$
$R(y) \longleftarrow R(x),\ y = x + 1.$

One approach to this problem could be to look for useful restrictions on the spatial databases under consideration, or on the syntax of allowed programs, guaranteeing termination. In the present paper we demonstrate through a series of examples that such restrictions are unlikely to exist. For example, we will see that even under severe restrictions such as the following, programs may still not terminate:

1. Only bounded spatial data are considered;
2. No arithmetic at all is allowed in programs;
3. Rules can only be of the form

$$R(x, y) \longleftarrow S(x, y),\ B,$$

where S is the spatial database predicate and B is the rest of the rule body. In other words, it is syntactically guaranteed that derived relations in the program can only contain points from the given bounded data set.

Our observations stand in contrast to results achieved by Revesz and others [8, 13, 14], which show that in other, non-spatial, instances of the constraint database framework (more specifically, databases and queries involving order constraints only), useful termination guarantees can be found.

We conclude that the termination of particular spatial recursive queries will have to be established by ad-hoc arguments, if it can be established at all. As a case study we investigate the topological connectivity query mentioned earlier. We give a program in the spatial version of Datalog that expresses the connectivity query in a natural way. It can be easily shown to terminate on *linear* spatial databases but the program may loop forever on non-linear spatial databases, even bounded ones. We can, however, give an effective characterization of the class of semi-algebraic sets for which termination is guaranteed. The proof of termination relies on Collins's Cylindrical Algebraic Decomposition method [4, 1].

This paper is organized as follows. Some definitions are given in Sect. 2. A series of examples of non-terminating programs is developed in Sect. 3. The topological connectivity query is discussed in Sect. 4. Conclusions are presented in Sect. 5.

2 Preliminaries

In the present paper, we consider a *spatial database* to be a geometrical figure in the real plane. A geometrical figure in the plane is a possibly infinite set S of points in $\mathbf{R}^2$, where $\mathbf{R}$ stands for the real numbers. Equivalently, it is a binary relation on the infinite domain $\mathbf{R}$.

One wants almost always to represent a geometrical figure in some effective, finite manner. A rather general way of doing this is by using *real polynomial inequalities* of the form $p(x, y) > 0$, where $p(x, y)$ is a polynomial in the variables x and y with real coefficients. Such an inequality defines the figure $\{(x, y) \mid p(x, y) > 0\}$. By using Boolean combinations (union, intersection, and complement) of polynomial inequalities one can describe a rather general class of figures, known as the class of *semi-algebraic sets*. Note that $p(x, y) = 0$ can be expressed as $\neg(p(x, y) > 0) \wedge \neg(-p(x, y) > 0)$.

Example. As an example, the semi-algebraic set already mentioned in the Introduction is shown in Fig. 1. □

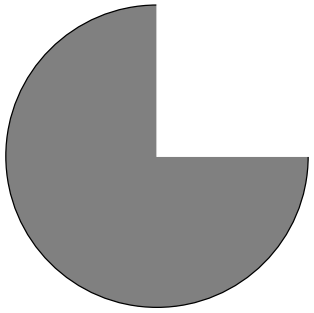

Fig. 1. Semi-algebraic set defined by $x^2 + y^2 \leq 1 \wedge \neg(x \geq 0 \wedge y \geq 0)$.

We assume familiarity with the language Datalog. We can turn Datalog into a spatial query language as follows:

- The underlying domain is the set of real numbers.

- The only EDB predicate is S, which is interpreted as the set of points in the spatial database, or equivalently, as a binary relation.
- Relations can be infinite.
- Polynomial inequalities are allowed in rule bodies.

Under the bottom-up semantics, the following fundamental closure property is satisfied by any Datalog program P [8]:

> *if the input relation S is semi-algebraic, then every derived relation R obtained by a finite number of iterations of P is also semi-algebraic; moreover, a finite representation of R can be effectively computed.*

Example. As a simple example, the following program derives in R the set of all points that either lie in the database or to the left of a point in the database:

$R(x, y) \longleftarrow S(x, y)$
$R(x, y) \longleftarrow R(x', y),\ x < x'.$

This program always terminates after two iterations and hence, by the above, when applied to a semi-algebraic set S, will produce a set R that is also semi-algebraic. For example, if S is the set of Fig. 1, then R will be the set

$$\{(x, y) \mid S(x, y) \vee (\exists x')(S(x', y) \wedge x < x')\},$$

which can be defined by

$$-1 \leq y < 0 \wedge (x^2 + y^2 \leq 1 \vee x < 0) \quad \vee \quad 0 \leq y \leq 1 \wedge x < 0.$$

□

The above example also illustrates that the effective computation of a semi-algebraic representation of the result involves the elimination of quantifiers; a classical theorem by Tarski [16] says that every logical description of a real figure with quantifiers can also be described without quantifiers. Good algorithms for quantifier elimination over the reals are known since the work of Collins [4, 1]; for an overview of the recent advances see Renegar's series of papers [12].

3 Non-terminating programs

In the Introduction we already mentioned the trivial program

$R(0) \longleftarrow$
$R(y) \longleftarrow R(x),\ y = x + 1.$

This program does not terminate, regardless of the input database (which is not used at all in the program), because the resulting set R grows unboundedly. Let us therefore see what happens if a fixed bound is put on the result, by requiring that the result must stay within some fixed bounded region.

The following example shows that under this requirement programs may still loop forever:

$R(1) \longleftarrow$
$R(x) \longleftarrow R(x'),\ x = x'/2.$

Relation R remains in the interval $[0, 1]$ but the program does not terminate; the interval is repeatedly cut in half and this goes on indefinitely.

In search of termination guarantees, we therefore put the drastic restriction on programs that no arithmetic (polynomial inequalities) is allowed in rule bodies. In other words, we concentrate on pure Datalog programs. Clearly, under this restriction, programs that do not rely on the database, such as the two programs given above, can no longer loop forever.

The classical example in pure Datalog is the transitive closure program TC:

$R(x, y) \longleftarrow S(x, y)$
$R(x, y) \longleftarrow R(x, z),\ S(z, y).$

On finite inputs S, TC always terminates. On infinite inputs, however, this is of course no longer true. For example, TC will loop forever on the semi-algebraic set S defined by $y = x + 1$; in the n-th iteration all points of the form $(x, x + n)$ are added.

It is a basic fact of standard relational database theory that on finite relations, pure Datalog programs always terminate [17]. One might hope that this property carries over to infinite but *bounded, semi-algebraic* sets. For example, by bounding the above-mentioned set $y = x + 1$ as

$$y = x + 1 \wedge 0 \leq x \leq 10,$$

TC will terminate after ten iterations. This hope is unjustified, however. On the bounded input

$$S = \{(x, y) \mid y = 2x \wedge 0 \leq x \leq 1/2\},$$

TC will loop forever; every iteration adds a line segment of increasing slope, as illustrated (for the first three iterations) in Fig. 2.

The previous example also shows that, even if we require not only the *input* set but also the *output* set to be bounded, programs still can loop forever. Indeed, the lines of increasing slope all remain in the unit square.

As a last resort in our search for termination guarantees, we therefore investigate the stronger requirement that the output set is always contained in the input set. In other words, we concentrate on *selection queries*. The output of a selection query on a bounded input is clearly also bounded.

Whether or not a program expresses a selection query is a semantic property, but we can put a strong syntactic range restriction on the rules so that only selection queries can be expressed: every rule has to be of the form

$$R(x, y) \longleftarrow S(x, y),\ B,$$

where S is as always the EDB predicate denoting the database and B is the rest of the rule body. A program in which every rule is of this form is called a *selection program*. The transitive-closure program is not a selection program.

The following program, a range-restricted variation of TC, is an example of a selection program:

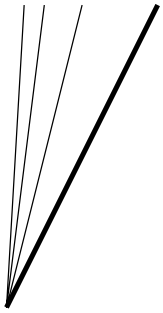

Fig. 2. First three iterations of the transitive closure of $S = \{(x, y) \mid y = 2x \wedge 0 \leq x \leq 1/2\}$.

$R(x, y) \longleftarrow S(x, y),\ S(x, x),\ S(y, y)$
$R(x, y) \longleftarrow S(x, y),\ R(z, x).$

On unbounded semi-algebraic inputs, selection programs can loop forever, as witnessed by the set

$$S = \{(x, y) \mid y = x + 1 \wedge x \geq 0\} \cup \{(0, 0)\}.$$

On this S, the above program will not terminate; in the first iteration the point $(0, 0)$ is added, and in subsequent iterations subsequent points of the form $(n, n+1)$ are added. However, if we bound S, e.g., by adding the constraint $y \leq 10$, the program will terminate.

Hence, our last hope is that

selection programs on bounded semi-algebraic inputs always terminate.

The following set destroys also this last hope:

$$S = \{(x, y) \mid y = x/2 \wedge 0 < x \leq 1\} \cup \{(1, 1)\}.$$

On this input, the above program does not terminate; in the first iteration the point $(1, 1)$ is added, in the second iteration the endpoint of the line segment in the database is added, and in subsequent iterations the line segment is repeatedly cut in half, going on indefinitely. The first four iterations are illustrated in Fig. 3.

Remark: We point out that all spatial databases considered in the above discussion are *linear*, in the sense that they can be defined using linear polynomials only.

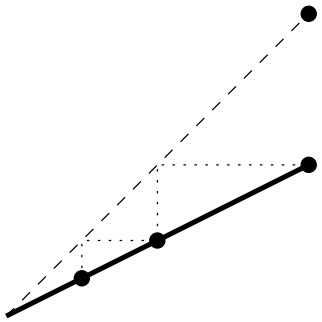

Fig. 3. First four iterations of a non-terminating selection program on a bounded semi-algebraic input.

4 Connectivity

A set S of points in the plane is called *connected* (in the sense of Topology) if it cannot be partitioned by two disjoint open sets. If S is semi-algebraic, S is connected if and only if any pair of points in S can be linked by a semi-algebraic curve lying entirely in S [3].

In this section we will present a program for testing connectivity. The program is written in Datalog with polynomial inequalities and stratified negation.[3] The program works correctly (in particular, terminates) on any linear semi-algebraic set. It may, however, not work correctly (in particular, loop forever) on other spatial databases, even on bounded ones. The main theorem of this section gives an effective characterization of the class of semi-algebraic sets on which the proposed program is guaranteed to terminate and produce a correct result.

The program is shown in Fig. 4. The first pairs of points in S that are derived by the program in relation *Path* are those that are connected by a straight line segment lying entirely in S. Then the transitive closure of *Path* is computed. After n iterations of the transitive closure, *Path* contains all pairs of points in S that can be connected by a piecewise linear curve consisting of n line segments lying entirely in S.

To prove correctness of this program on a class $\mathcal{C}$ of spatial databases we must establish two facts for every spatial database S in $\mathcal{C}$:

- *Soundness*: Two points in S are in the same connected component of S if and only if they can be connected by a piecewise linear curve lying entirely in S;
- *Termination*: The number of line segments needed to connect any such pair of points in S is bounded.

[3] We will refer to this language simply as "Datalog".

$$
\begin{aligned}
\mathit{Obstructed}(x,y,x',y') &\longleftarrow \neg S(\bar{x},\bar{y}),\ \bar{x} = ax + a'x',\ \bar{y} = ay + a'y',\\
&\qquad 0 \le a \le 1,\ 0 \le a' \le 1,\ a + a' = 1\\
\mathit{Path}(x,y,x',y') &\longleftarrow S(x,y),\ S(x',y'),\ \neg\, \mathit{Obstructed}(x,y,x',y')\\
\mathit{Path}(x,y,x',y') &\longleftarrow \mathit{Path}(x,y,x'',y''),\ \mathit{Path}(x'',y'',x',y')\\
\mathit{Disconnected} &\longleftarrow S(x,y),\ S(x',y'),\ \neg \mathit{Path}(x,y,x',y')\\
\mathit{Connected} &\longleftarrow \neg\, \mathit{Disconnected}.
\end{aligned}
$$

Fig. 4. A Datalog program for topological connectivity.

Termination guarantees that the transitive closure will terminate. Soundness then establishes the correctness of the test for connectivity performed by the program after the transitive closure is completed.

Example. Linear semi-algebraic sets always satisfy soundness and termination. This will be a corollary of our main theorem.

The set defined by $(y-x^2)(x^2-y+1/2) > 0$, shown in Fig. 5, does not satisfy termination. There is no upper bound on the number of line segment needed to connect the point at the origin to other points in the set; the higher the other point, the more line segments are needed.

Even on bounded semi-algebraic sets correctness cannot be guaranteed. Here we take as example the set defined by $y^3 - x^2 \ge 0 \wedge x \le 0 \wedge y \le 1$, depicted in Fig. 6. Indeed, since the tangent to the curve in the bottom point coincides with the vertical line, any straight line segment from the interior of the database (indicated as the shaded area) to the bottom point will leave the interior of the database. This implies that both soundness and termination fail in this case. □

Fig. 5. A database on which termination is not satisfied. The database consists of the points lying strictly between the parabola $y = x^2$ and the translated one $y = x^2 + 1/2$.

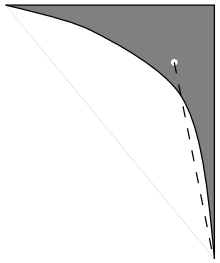

Fig. 6. Both soundness and termination fail on this bounded database.

We will actually demonstrate that the above examples of Figs. 5 and 6 illustrate essentially the only two cases on which the program of Fig. 4 will not work correctly. Our development will proceed in two steps:

1. First, we show how we can derive from a given semi-algebraic set S another semi-algebraic set S' in Datalog. The set S' is special in that it will only contain "two-dimensional" points (defined precisely below). Furthermore, S' is connected if and only if S is.
2. By the first step, we can then in a second step restrict our attention to sets containing only two-dimensional points, and give a necessary and sufficient condition on these sets for correctness and termination of the Datalog program of Fig. 4.

The notion of 2-dimensional point depends on the notion of "observation" of a point with respect to some semi-algebraic set. This is formalized in the following definition. For a point p and an $\varepsilon > 0$, let $C(p, \varepsilon)$ denote the circle with radius ε and center p and $D(p, \varepsilon)$ the closed disk with radius ε and center p.

Definition. Let S be a semi-algebraic set and let p be a point.

- An $\varepsilon > 0$ is called an *S-observation radius* of p if for each ε' such that $0 < \varepsilon' < \varepsilon$, $S \cap C(p, \varepsilon)$ is isotopic to $S \cap C(p, \varepsilon')$.
- An *S-observation circle* of p is a circle around p whose radius is an S-observation radius of p.
- The minimum of 1 and the supremum of all S-observation radii of p is denoted by $\varepsilon_{p,S}$.

Known properties of semi-algebraic sets [5] guarantee that these notions are well-defined.

Using the notion of observation we can introduce three classes of points with respect to S as follows:

Definition. A point is

- 0-dimensional if its S-observation circles do not intersect S;
- 1-dimensional if each S-observation circle intersects S in two points;
- 2-dimensional if each S-observation circle intersects S in infinitely many points.

The points in S that are 0-dimensional with respect to S are the isolated points of S. The 1-dimensional points in S are those lying locally on a curve of S. The 2-dimensional points with respect to S are those for which the intersection of even the smallest of their neighborhoods with S has a non-empty interior. Remark that there are points in the plane which do not belong to any of these classes.

For each semi-algebraic set in $\mathbf{R}^2$ we now define its "blown-up" version. An illustration is given in Fig. 7.

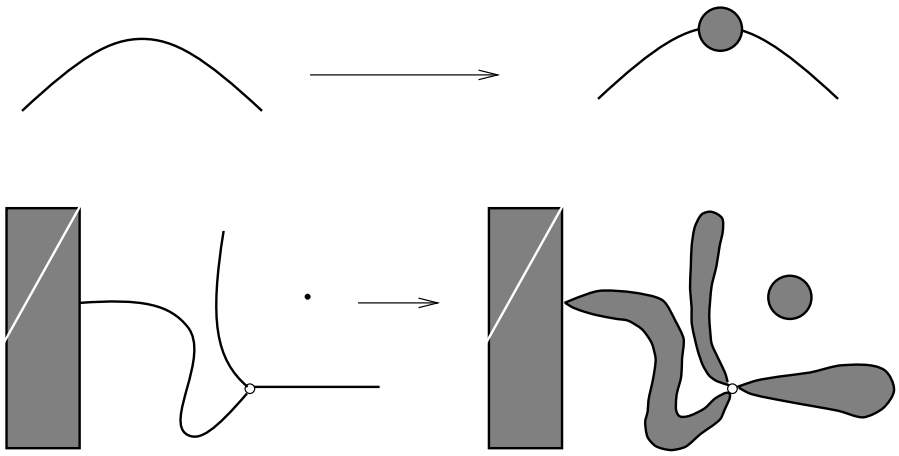

Fig. 7. A semi-algebraic set and an approximated picture of its blown-up version.

Definition. Let S be a semi-algebraic set. The *blown-up version of* S equals

$$BU(S) = S \cup \bigcup_{p \in S_{01}} D(p, \varepsilon_{p,S}/3),$$

where S_{01} is the set of 0- and 1-dimensional points of S.

The proof of the following proposition is straightforward.

Proposition. *(i) $BU(S)$ can be computed by a recursion-free Datalog program, and hence is semi-algebraic.*

(ii) All points in $BU(S)$ are 2-dimensional with respect to $BU(S)$.
(iii) $BU(S)$ is connected if and only if S is.

A final technical concept we need is that a semi-algebraic set S is "conical" locally around each of its points. More specifically, if $p \in S$, then $D(p, \varepsilon_{p,S}) \cap S$ is homeomorphic to a planar cone with top p and base $C(p, \varepsilon_{p,S}) \cap S$. We will call the 2-dimensional regions of $D(p, \varepsilon_{p,S}) \cap S$ the *sectors of S around p*. Fig. 8 gives an example of a point with three sectors.

Analogously, there exists a $\varepsilon_{\infty,S} > 0$ such that $\{(x, y) \mid x^2 + y^2 \geq \varepsilon_{\infty,S}\} \cap S$ is homeomorphic to the unbounded planar cone with top at infinity defined as

$$\{(\lambda x, \lambda y) \mid (x, y) \in S \wedge x^2 + y^2 = \varepsilon_{\infty,S} \wedge \lambda \geq 1\}.$$

Here, we call the 2-dimensional regions of $\{(x, y) \mid x^2 + y^2 \geq \varepsilon_{\infty,S}\} \cap S$ the *unbounded sectors of S*.

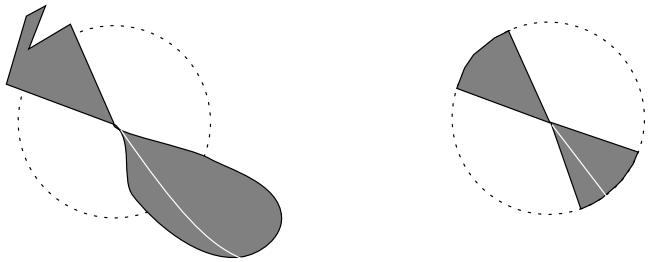

Fig. 8. The sectors of a semi-algebraic set around a point and the corresponding cone.

After the following important definition we will be ready to state and prove the announced characterization.

Definition. Let S be a semi-algebraic set. A semi-algebraic set S is called *border-visible* if

1. for each point p on the border of S and each sector of S around p there exists a interior point q of S in that sector such that the half open line segment $(p, q]$ is contained in the interior of S, and
2. for each unbounded sector of S there exists an unbounded half line completely contained in that region.

Example. The set of Fig. 6 is not border-visible because it does not satisfy the first condition of the definition; the set of Fig. 5 is not border-visible because it does not satisfy the second condition. □

Theorem. *Let S be a semi-algebraic set that contains only 2-dimensional points. The Datalog program of Fig. 4 tests correctly for topological connectivity of S in finite time, if and only if S is border-visible.*

Proof. First, assume S is border-visible. According to the discussion in the beginning of this Section, we need to establish soundness and termination.

A. Soundness. It suffices to show soundness in the interior only, i.e., to show that two points in S°, the topological interior of S, are in the same connected component of S° if and only if they can be connected by a piecewise linear curve lying entirely in S°. Indeed, because S is two-dimensional, the points in S on the border of S are also on the border of S°, and thus soundness for points on the border follows from soundness in the interior by border-visibility.

The if-implication is trivial. So we focus on the only-if implication. Two interior points that belong to the same connected component can be connected by a semi-algebraic curve lying entirely in the S°. It is well known that a uniformly continuous curve (such as a semi-algebraic one) can be arbitrarily closely approximated by a piecewise linear curve [9].

B. Termination. To establish termination we use a refined version of Collins's Cylindrical Algebraic Decomposition (CAD) [4, 1]. Collins's CAD, when applied to a semi-algebraic set S, shows the existence of a decomposition of S in a finite number of cells, where each cell is either a *point*, a 1-dimensional *curve* or a 2-dimensional *region*. We can refine this decomposition in the extrema and points of inflection of the upper and lower borders of the cells, as illustrated for a bounded region in Fig. 9. This yields a cell decomposition in which each cell has an upper and lower border which is constant, monotonic concave or monotonic convex. We can further adapt the decomposition, by merging, wherever possible, any two adjacent cells where one lies above the other.

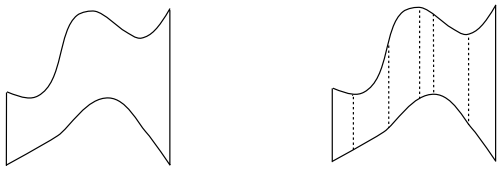

Fig. 9. An example of a region in Collins's CAD and its refined version.

Since the decomposition is finite, it suffices to show for each cell C that there exists an upper bound $\alpha(C)$ on the number of line segments needed in a piecewise linear path between any two points in C. This is trivial for those cells that are vertical line segments or single points.

The most difficult case is that where C is a bounded region. This situation is illustrated in Fig. 10. Because of the merging of adjacent cells mentioned above, the left bottom point p of the region is on the border of S. Let T be the sector around p that intersects C. Because S is border-visible there exists a point r

in T° such that the line segment $(p, r]$ is contained in T°. This point r is not necessarily in C°. Therefore, we also choose a point s in $C^\circ \cap T^\circ$.

We can perform a symmetric construction for the right upper point p', leading to sector T' and points r' and s'. By the reasoning already employed in the soundness part of this proof, there is a piecewise linear curve going from r to s lying entirely in T°, another going from r' to s' lying entirely in $(T')^\circ$, and another going from s to s' lying entirely in C°. Hence, there is a piecewise linear curve γ going from p to p' lying entirely in $C^\circ \cup T^\circ \cup (T')^\circ$.

Let $\beta(C)$ be the number of segments of γ. We can now take $\alpha(C) = \beta(C)+2$. Indeed, let q and q' be points in C. Because γ transverses C completely, there are points on γ with the same x-coordinate as q and q'. We can connect q and q' with these points using a single vertical line segment lying in C (indicated in Fig. 10 by dashed lines). So, q and q' can be connected by a piece-wise linear curve consisting of at most $\beta(C) + 2$ line segments.

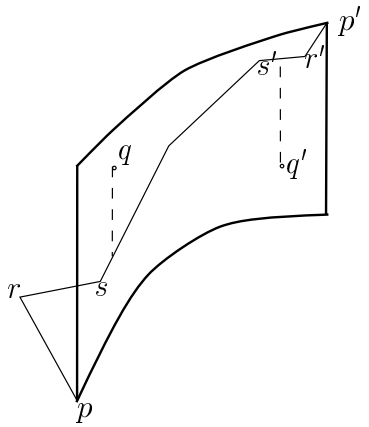

Fig. 10. The α of a bounded region.

The case where C is an unbounded region is analogous. By border-visibility, we know that there exist unbounded lines starting in a point from the interior of C and going into the direction where C is of unbounded. One can think of the points p, r and s (or p', r' and s') as all coinciding at infinity.

Finally, we prove the only-if implication of the theorem. Thereto, assume S is not border-visible. So, at least one of the following two possibilities arises:

1. there exists a border-point p of S and a sector T around p such that there is no point in the interior of T for which the line segment $(p, q]$ is contained in the interior of S,
2. there is an unbounded sector of S in which no half line is contained.

Using the CAD of above, we see that these two possibilities are represented by the examples we have seen in Figs. 5 and 6. We already argued informally that in both cases, termination fails. It is a standard mathematical exercise to make these arguments formal. □

Corollary. *The Datalog program of Fig. 4 tests correctly for topological connectivity of S in finite time on every linear semi-algebraic set.*

Proof. Linear sets are clearly border-visible. The 2-dimensionality assumption in the theorem is only used in the soundness argument to deal with border points; in a linear set this is not necessary since its border is itself piecewise linear. In other words, blowing up is unnecessary in the linear case. □

We show that the class of semi-algebraic sets for which the Datalog program of Fig. 4 correctly tests connectivity is a decidable class.

Theorem. *It is decidable whether a given semi-algebraic set that consists of 2-dimensional points is border-visible.*

Proof. We briefly sketch a decision algorithm. Let S be a semi-algebraic set that consists of 2-dimensional points. The border-points of S can be classified according to the type of cone (i.e., the configuration of sectors around them) they have. The set of border-points of S that have exactly one sector in their cone can be expressed in the relational calculus with polynomial inequalities by a formula that selects those border-points of S for which even the smallest circle around it has one single (closed, open, or half-open/half-closed) arc segment in common with S. Although there is an uncountably infinite number of these points, the condition of border-visibility for these points can be translated into the relational calculus with polynomial inequalities, and we have noted in the Introduction that this calculus is effective.

It can be easily shown that the number of border-points of S with more than one sector in their cone is finite. To check the condition of border-visibility for these points, we loop through all possible cone types with more than one sector and for each cone type we select the border-points of S that have a cone of that type. There are symbolic algorithms for the first-order theory of the reals (e.g., Collins's algorithm [1, 4]) that can effectively enumerate these points.

Since we know, in this loop, the cone type for each such point p it is possible to express $\varepsilon_{p,S}$ in the relational calculus with polynomial inequalities and thus to compute it. Without knowing the particular cone type the latter is not possible. We can also compute the sectors around p by computing the connected components of $(D(p, \varepsilon_{p,S}) \cap S) \setminus \{p\}$. The computation of the connected components of a semi-algebraic set is explained in [15]. For each sector the condition of border-visibility can again be translated into the relational calculus with polynomial inequalities. We exit the loop when all border-points of S are processed. Since the border-points of S with more than one sector in their cone are finite in number, we exit the loop after a finite number of cone types have been considered.

Finally, for the unbounded sectors, we again have to loop through all possible cone types to determine the configuration of the unbounded sectors of S. Once we know this configuration, we can compute $\varepsilon_{\infty,S}$. We can compute the unbounded sectors by computing the connected components of $S \cap \{(\lambda x, \lambda y) \mid x^2 + y^2 = \varepsilon_{\infty,S} \wedge \lambda \geq 1\}$. The condition of border-visibility for the unbounded sectors can again be translated into the relational calculus with polynomial inequalities. □

5 Conclusion

It is not obvious how one can test for topological connectivity in the case of general semi-algebraic spatial databases by a program that always terminates. One may wonder whether topological connectivity is expressible at all in spatial Datalog. The query is certainly known to be algorithmically decidable in the general semi-algebraic case [15]. In this respect it should be noted that extending the relational calculus with polynomial constraints and while-loops yields a computationally complete query language for semi-algebraic spatial databases [7], and therefore connectivity is expressible in that language. It seems possible to simulate while-loops using an inflationary version of spatial Datalog. The problem remains however whether there exists a natural spatial query language in which connectivity is effectively expressible in a natural manner.

Acknowledgment

We are indebted to Rudi Penne for most helpful discussions on the issue of termination.

References

1. D.S. Arnon. Geometric reasoning with logic and algebra. *Artificial Intelligence*, 37:37–60, 1988.
2. M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. In *Proceedings 15th ACM Symposium on Principles of Database Systems*. ACM Press, 1996.
3. J. Bochnak, M. Coste, and M.-F. Roy. *Géométrie algébrique réelle*. Springer-Verlag, 1987.
4. G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Lecture Notes in Computer Science*, 33:134–183, 1975.
5. M. Coste. Ensembles semi-algébriques. In *Géometrie algébrique réelle et formes quadratiques*, volume 959 of *Lecture Notes in Mathematics*, pages 109–138. Springer, 1982.
6. S. Grumbach and J. Su. First-order definability over constraint databases. In Montanari and Rossi [10], pages 121–136.
7. M. Gyssens, J. Paredaens, J. Van den Bussche, and D. Van Gucht. Computable queries for spatial database systems. In preparation.
8. P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

9. E.E. Moise. *Geometric topology in dimensions 2 and 3*, volume 47 of *Graduate Texts in Mathematics*. Springer, 1977.
10. U. Montanari and F. Rossi, editors. *Principles and practice of constraint programming*, volume 976 of *Lecture Notes in Computer Science*. Springer, 1995.
11. J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings 13th ACM Symposium on Principles of Database Systems*, pages 279–288. ACM Press, 1994.
12. J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13, 1992.
13. P.Z. Revesz. A closed-form evaluation for Datalog queries with integer (gap)-order constraints. *Theoretical Computer Science*, 116:117–149, 1993.
14. P.Z. Revesz. Safe stratified Datalog with integer order programs. In Montanari and Rossi [10], pages 154–169.
15. 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.
16. A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.
17. J. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.