*Proc. Int. Workshop on Database Programming Languages (DBPL), 1999*

# A Representation Independent Language for Planar Spatial Databases with Euclidean Distance

Gabriel M. Kuper[1] and Jianwen Su[2]

[1] Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974.
kuper@research.bell-labs.com
[2] Department of Computer Science, University of California, Santa Barbara, CA 93106. su@cs.ucsb.edu

**Abstract.** Linear constraint databases and query languages are appropriate for spatial database applications. Not only the data model is natural to represent a large portion of spatial data such as in GIS systems, but also there exist efficient algorithms for the core operations in the query languages. However, an important limitation of the linear constraint data model is that it cannot model constructs such as "Euclidean distance." A previous attempt to expend linear constraint languages with the ability to express Euclidean distance, by Kuijpers, Kuper, Paredaens, and Vandeurzen is to adapt two fundamental Euclidean constructions with ruler and compass in a first order logic over points. The language, however, requires the input database to be encoded in an ad hoc LPC representation so that the logic operations can apply. This causes a problem that sometimes queries in their language may depend on the encoding and thus do not have any natural meaning. In this paper, we propose an alternative approach and develop an algebraic language in which the traditional operators and Euclidean constructions work directly on the data represented by "semi-circular" constraints. By avoiding the encoding step, our language do not suffer from this problem. We show that the language is closed under these operations.

## 1 Introduction

First-order logic with linear constraints (FO+lin) has turned out to be an appropriate language for expressing queries over spatial data, as in GIS systems, for example. There are, however, certain limitations on the expressive power of FO+lin. Some of these limitations are inherent to first-order languages in general, including the fact that connectivity cannot be expressed, and the tradeoffs between expressive power and efficiency in such cases have been well studied [BDLW98,PVV98,GS97]. There are, however, additional limitations that are a result not of the language itself, but rather of the class of linear constraints. The most significant of these restrictions is the inability to express the notion of Euclidean distance.

If we were to consider a query language with polynomial constraints (FO+poly), we would clearly be able to express such queries, but such a lan-

guage would be far too powerful for our purposes and would be more difficult to implement. Although such a language is theoretically feasible, practical algorithms for efficient implementation of database systems with FO+poly are still a research issue. A natural question is to ask whether there is a language between FO+lin and FO+poly with additional expressive power, but without the full power of FO+poly. The naive approach, restricting our attention to quadratic constraints, does not work—the requirement that the language be closed enables one to write queries whose results require higher-order polynomials. In addition, adding some geometric primitives, such as collinearity, to a first-order language, again yields the full power of FO+poly. A more successful approach is the PFOL language of [VGV98]. This language enables one to express queries on finite databases that use Euclidean distance. However, as long as one restricts the attention to databases in FO+lin, one will still not be able to deal with queries such as "return all the points within a given distance," over finitely representable databases.

In DBPL '97, a first attempt was made at a different approach to this problem [KKPV97]. The key observation used there is that the two relevant concepts, linear constraints and Euclidean distance, correspond to the two basic operations of Euclidean geometry: constructions with ruler and compass. [KKPV97] defines a query language $\Phi_{\mathsf{circ}}$ that is situated strictly between FO+lin and FO+poly and that expresses those queries that can be described as ruler-and-compass constructions on the input database.

The original idea in the work of Kuijpers et al [KKPV97] had been to use lines and circles as primitive objects, and define operations on them. This did not work, as Euclidean geometry provided a clear intuition for what to do with these lines and circles, but not about interiors of regions – in other words, there was no natural way to define the operations on interiors of a region that were naturally derived from operations on their boundaries. For this reason, $\Phi_{\mathsf{circ}}$ applied to an encoding of objects as tuples of points. Using this encoding, $\Phi_{\mathsf{circ}}$ had the desired properties.

Since objects in constraint databases are not encoded, a $\Phi_{\mathsf{circ}}$ query consists of three parts: an encoding step, that maps a flat relation to its encoding, the "real" query, that works on this encoding, and a decoding step. The semantics of the query language depends on a specific, but arbitrary, encoding and this causes certain problems in $\Phi_{\mathsf{circ}}$. Indeed the query language allows queries with no natural meaning ("return the object whose representative is closest to the origin") to be expressed.

In this paper we propose a different approach in which the data is represented directly as spatial objects. The model is reminiscent of nested extension [GS95] of the standard constraint model. The purpose is to avoid the need to use an encoding to refer to distinct geometrical objects. Our main contribution is to provide a natural extension of standard Euclidean operations to *interiors* of regions. We generalize the notion of drawing a line (or

circle) between 2 points to that of drawing a line between 2 objects. The idea is that, given two objects, we take the union of all the lines that go through pairs of points from the two given objects. (A similar idea, drawing lines through the origin, was used in [VGV98], but only for linear databases). This may appear to give no additional power: as we shall see, the result can always be described by taking the boundary of the objects, drawing the appropriate lines between boundary points, and taking the interior of the result. Our direct approach, however, has the advantage of establishing a *direct*, natural connection between the original objects and the result of the operation, thus eliminating the need for auxiliary information to specify which interiors are in the database.

In the next section we give basic definitions, and the following section defines the Euclidean operations on regions and the EuAlg languages based on them. We then prove that the language is closed, and conclude with related work and directions for future research.

## 2 Basic Notions

We consider spatial databases in the plane. In order to accommodate Euclidean operations, these must be over a subfield $\mathbb{D}$ of $\mathbb{R}$ that is closed under square roots. Most of our results apply to any such field. The minimal such field is known as the field of *constructible numbers*. As in [KKPV97], we consider sets of points that can be described by lines and circles. These are called *semi-circular sets*.

**Definition.** A subset of $\mathbb{D}^2$ is called a *semi-circular set* iff it can be defined as a Boolean combination of sets of the form

$$\{(x, y) \mid ax + by + c\ \theta\ 0\}$$

or

$$\{(x, y) \mid (x - a)^2 + (y - b)^2\ \theta\ c^2\}\ ,$$

where $a, b$, and $c$ are in the domain $\mathbb{D}$ and $\theta$ in $\{\leq, <, =, \geq, >\}$. Let $\boldsymbol{P}$ be the set of all semi-circular sets over $\mathbb{D}^2$.

**Definition.** A *Euclidean constraint* is an equation in one of the following two forms:

$$ax + by + c = 0$$

or

$$(x - a)^2 + (y - b)^2 = c^2\ ,$$

with $a$, $b$, and $c$ in the domain $\mathbb{D}$.

A semi-circular set is called *rectangular* if it can be represented by a formula of the form $a < x < b \wedge c < y < d$, with $a$, $b$, $c$, $d \in \mathbb{D}$.

**Definition.** Let $r$ be a semi-circular set.

1. The *boundary* of $r$ is the set of all points $p$ in in $\mathbb{D}^2$, such that every non-empty rectangular set that contains $p$ contains both points in $r$ and points not in $r$.
2. A *side* of $r$ is a maximal set of all those boundary points that satisfy a single Euclidean constraint.
3. A point $p$ in $r$ is an *isolated point* of $r$ if there is a non-empty rectangular set that contains $p$, but contains no other point of $r$.
4. A point $p$ in $\mathbb{D}^2$ is a *corner* of a region $r$ if $p$ is either (1) an isolated point of $r$, or (2) a boundary point of $r$ that is a member of at least two sides of $r$.

Note that the notions of sides and corners are defined in such a way to generalize the intuitive notion of a "side" of a semi-linear set to include segments of circles as well as straight lines. It is straightforward to show (1) that any semi-circular set has a finite number of sides, (2) that each side of a semi-circular set is itself semi-circular set, and (3) that the boundary of a semi-circular set is also a semi-circular set. Note that the definitions apply to unbounded sets as well; in particular, $\mathbb{D}^2$ has no sides and has empty boundary.

## 3 The EuAlg Language

We first define the data model. The basic types are 2-dimensional semi-circular sets. We shall use the term *semi-circular relation* for relations over these types (our terminology here is different from that used in [KKPV97], which uses a flat model, where relations are simply semi-circular sets).

**Definition.** A *semi-circular $n$-tuple* is a tuple $t = (t_1, \ldots, t_n)$, where each $t_i$ is a semi-circular set. Two tuples $t$ and $t'$ are *equivalent* if the semi-circular sets represented by $t_1, \ldots, t_n$ are equal to the semi-circular sets represented by $t'_1, \ldots, t'_n$, respectively. A *semi-circular relation $R$ of arity $n$* is a finite set of semi-circular $n$-tuples.

Equivalence and containment of semi-circular relations can now be defined in a natural way; these are decidable, which follows from the decidability of the theory of real closed fields. Note that equivalence of relations differs from equivalence in the sense of [KKR95], as the current model is a nested one. In the current paper, we ignore non-spatial (thematic attributes), though these can be easily added to the model.

EuAlg is a nested algebraic query language, but with only one level of nesting (this is just to provide names for spatial objects). The nested model itself similar to the ones used in [BBC97], as well as the Dedale [GRS98] and Cosmos [KRSS98] prototypes.

We now turn to the spatial primitives. The novel ones are extensions of the Euclidean primitives for drawing lines and circles to handle regions. We start with lines. The intuition is that the generalization of the notion of drawing a line between two points, to that of drawing lines between two regions, is to take the union of all lines that go through any point in the first region and any point in the second. For technical reasons, we actually use rays, rather than lines, a ray from $p_1$ to $p_2$ being a "half-line," starting at $p_1$, and going through $p_2$ (a line is then easily definable as the union of two rays).

How do we handle circles? A circle in [KKPV97] is represented by a triple $(p_1, p_2, p_3)$, where $p_1$ is the center, and $d(p_2, p_3)$ is the radius. We generalize this directly to semi-circular sets, by taking the union of all circles with center in the first set, and radius equal to the distance between a point in the second and one in the third. (Alternative approaches are discussed in Section 5).

**Definition.** Let $p_1$, $p_2$, and $p_3$ be points in $\mathbb{D}^2$.

1. $ray(p_1, p_2)$ is the half line that starts at $p_1$ (including $p_1$ itself)and goes through the point $p_2$.
2. $circ(p_1, p_2, p_3)$ is the circle with center $p_1$, and radius equal to $d(p_2, p_3)$.

**Definition.** Let $r_1$, $r_2$, and $r_3$ be semi-circular sets.

1. $\textsc{ray}(r_1, r_2) = \bigcup_{p_1 \in r_1, p_2 \in r_2} ray(p_1, p_2)$.
2. $\textsc{circ}(r_1, r_2, r_3) = \bigcup_{p_1 \in r_1, p_2 \in r_2, p_3 \in r_3} circ(p_1, p_2, p_3)$.
3. $\textsc{bdr}(r_1) = \{p \mid p \text{ is a boundary point of } r_1\}$.
4. $\textsc{sides}(r_1) = \{r' \mid r' \text{ is a side of } r_1\}$.
5. $\textsc{corners}(r_1) = \{p \mid p \text{ is a corner of } r_1\}$.

**Example 1.** *Consider the two regions $r_1$ and $r_2$, where*

$$r_1 = 1 \leq x \leq 2 \wedge 1 \leq y \leq 2$$

*and*

$$r_2 = 3 \leq x \leq 4 \wedge 2 \leq y \leq 3 \ ,$$

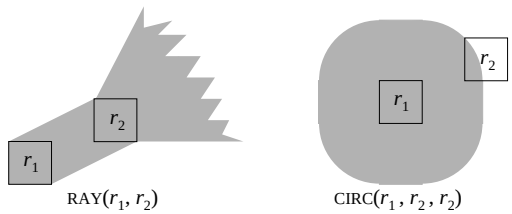


**Fig. 1.** Ray and circle drawing on two rectangles

*Then*

$$
\begin{aligned}
\textsc{ray}(r_1, r_2) &= s_1 = (0 \le 2y - x \le 3 \wedge x \ge 1 \wedge y \ge 1) \\
&\quad \vee (y - 2x \le 3 \wedge y \ge 2) \\
\textsc{ray}(r_1, r_1) &= s_2 = \mathbb{D}^2 \\
\textsc{circ}(r_1, r_2, r_2) &= s_3 = (1-\sqrt{2} \le x \le 2+\sqrt{2} \wedge 1 \le y \le 2) \\
&\quad \vee (1 \le x \le 2 \wedge 1-\sqrt{2} \le y \le 2+\sqrt{2}) \\
&\quad \vee ((x-1)^2 + (y-1)^2 \le 2) \\
&\quad \vee ((x-1)^2 + (y-2)^2 \le 2) \\
&\quad \vee ((x-2)^2 + (y-1)^2 \le 2) \\
&\quad \vee ((x-2)^2 + (y-2)^2 \le 2) \\
\textsc{circ}(r_1, r_1, r_2) &= s_4 = (1-\sqrt{13} \le x \le 2+\sqrt{13} \wedge 1 \le y \le 2) \\
&\quad \vee (1 \le x \le 2 \wedge 1-\sqrt{13} \le y \le 2+\sqrt{13}) \\
&\quad \vee ((x-1)^2 + (y-1)^2 \le 13) \\
&\quad \vee ((x-1)^2 + (y-2)^2 \le 13) \\
&\quad \vee ((x-2)^2 + (y-1)^2 \le 13) \\
&\quad \vee ((x-2)^2 + (y-2)^2 \le 13) \\
\textsc{sides}(r_1) &= \{r_{1,1}, r_{1,2}, r_{1,3}, r_{1,4}\}\ , \\
&\quad \textit{where} \quad r_{1,1} = x = 1 \wedge 1 \le y \le 2 \\
&\qquad\qquad\quad r_{1,2} = x = 2 \wedge 1 \le y \le 2 \\
&\qquad\qquad\quad r_{1,3} = 1 \le x \le 2 \wedge y = 1 \\
&\qquad\qquad\quad r_{1,4} = 1 \le x \le 2 \wedge y = 2 \\
\textsc{bdr}(r_1) &= r_{1,1} \vee r_{1,2} \vee r_{1,3} \vee r_{1,4} \\
\textsc{corners}(r_1) &= \{(1,1), (2,1), (1,2), (2,2)\}
\end{aligned}
$$

*Figure 1 shows the result of* $\textsc{ray}(r_1, r_2)$ *and* $\textsc{circ}(r_1, r_2, r_2)$. ∎

The EuAlg algebra is standard relational algebra, together with special Euclidean operators. We start with the standard part:

**Definition.**

1. $r \cup s$ is the union of the relations $r$ and $s$, i.e., the union of the sets of tuples in both relations with duplicates (i.e., equivalent tuples) eliminated.
2. $r \cap s$ is the set of those tuples in $r$ that are equivalent to some tuple in $s$.
3. $r - s$ is the set of those tuples in $r$ that are not equivalent to any tuple in $s$.

4. $r \times s$ is the Cartesian product of $r$ and $s$.
5. $\sigma_F(r)$ where $F$ is one of $i = j$, $i \subseteq j$, $i \cap j = \emptyset$ (where $r$ has arity $n$, and $i$, $j \leq n$) is the set of those tuples in $r$ for which the sets in columns $i$ and $j$ satisfy $F$.
6. $\pi_X r$ is the set of all tuples of the for $\pi_X(t)$ for $t \in r$.

We now turn to the Euclidean operators. These operators have a certain resemblance to aggregate operators or functions in the relational model, in that each operators adds an additional column to the relation, whose value depends on the values of the other columns of each tuple. In the following definition, $r$ will be a relation of arity $n$, and $i$, $j$ and $k \leq n$. The result of $\mathcal{E}_{\text{OP}}(r)$ will be a relation of arity $n+1$, defined as follows:

**Definition.**

1. Set operators on the spatial extent:
   - Union: $\mathcal{E}_{i \cup j}(r) = \{(t, t.i \cup t.j) \mid t \in r\}$.
   - Intersection: $\mathcal{E}_{i \cap j}(r) = \{(t, t.i \cap t.j) \mid t \in r\}$.
   - Difference: $\mathcal{E}_{i-j}(r) = \{(t, t.i - t.j) \mid t \in r\}$.
2. $\mathcal{E}_{\text{RAY}(i,j)}(r) = \{(t, \text{RAY}(t.i, t.j)) \mid t \in r\}$.
3. $\mathcal{E}_{\text{CIRC}(i,j,k)}(r) = \{(t, \text{CIRC}(t.i, t.j, t.k)) \mid t \in r\}$.
4. $\mathcal{E}_{\text{BDR}(i)}(r) = \{(t, \text{BDR}(t.i)) \mid t \in r\}$.
5. $\mathcal{E}_{\text{SIDES}(i)}(r) = \{(t, s) \mid t \in r, s \in \text{SIDES}(t.i)\}$.
6. $\mathcal{E}_{\text{CORNERS}(i)}(r) = \{(t, s) \mid t \in r, s \in \text{CORNERS}(t.i)\}$.

Finally, the EuAlg language also has two constant relations $e_o$ and $e_u$ (for "origin" and "unit), that contain the tuples $\{(0,0)\}$ and $\{(0,1)\}$ respectively. The need for 2 fixed points was discussed in [KKPV97]: These points can be used to simulate choice constructs ("select an arbitrary point on a line"), that are used in many geometric constructions.

**Example 2.** *Consider the relations*

$$r = \{(r_1, r_2), (r_1, r_3)\}$$

*and*

$$s = \{(r_1, r_2, r_2), (r_1, r_1, r_2)\} \ ,$$

*where*

$$\begin{aligned} r_1 &= 1 \leq x \leq 2 \wedge 1 \leq y \leq 2 \\ r_2 &= 3 \leq x \leq 4 \wedge 2 \leq y \leq 3 \\ r_3 &= y = x + 4 \end{aligned}$$

*Then*

1. $\mathcal{E}_{\text{RAY}(1,2)}(r) = \{(r_1, r_2, s_1), (r_1, r_3, s_5)\}$, *where* $s_1 = \text{RAY}(r_1, r_2)$ *was described in Example 1 and* $s_5 = y > x - 1$.
2. $\mathcal{E}_{\text{CIRC}(1,2,3)}(s) = \{(r_1, r_2, r_2, s_3), (r_1, r_1, r_2, s_4)\}$ *where* $s_3 = \text{CIRC}(r_1, r_2, r_2)$ *and* $s_4 = \text{CIRC}(r_1, r_1, r_2)$ *were also described in Example 1.*

**Example 3.** *We now illustrate how Euclidean constructions can be expressed in* EuAlg. *Let* $r$ *be a binary relation that represents a set of pairs of lines. More formally, if* $(l_1, l_2)$ *in a tuple in* $r$*, then each* $l_i$ *represents a line. Suppose that we want to bisect the angles defined by these pairs of lines, i.e., to compute a relations* $s$ *such that* $(l_1, l_2, l)$ *is in* $s$ *iff* $(l_1, l_2)$ *is in* $r$ *and* $l$ *is the line that bisecting the angle from* $l_1$ *to* $l_2$*. We can express this as follows:*

1. *Compute the intersection of each pair of lines:*

$$r_1 = \mathcal{E}_{1\cap 2}(r) \ .$$

2. *Draw all circles with centers at these intersection points, and with radius equal to unity. Then take the intersections of these circles with the original lines:*

$$r_2 = \mathcal{E}_{2\cap 6}\mathcal{E}_{1\cap 6}\mathcal{E}_{\text{CIRC}(3,4,5)} r_1 \times e_o \times e_u \ .$$

3. *Draw the two circles with centers at these intersections, and with radii equal to the distance between them:*

$$r_3 = \mathcal{E}_{\text{CIRC}(7,7,8)}\mathcal{E}_{\text{CIRC}(8,7,8)} r_2 \ .$$

4. *Take the intersections of these circles, and then draw the rays through these points and through the vertex of the angle (the entire line is thus computed (note that each line is computed twice, but that duplicates are automatically eliminated). Finally the intermediate results are projected out:*

$$r_4 = \pi_{(1,2,12)}\mathcal{E}_{\text{RAY}(3,11)}\mathcal{E}_{9\cap 10} r_3 \ .$$

## 4 Closure

Our main result is that the EuAlg is closed:

**Theorem 1.** *Let* $Q$ *be a* EuAlg *expression, and* $r$ *a semi-circular relation. The* $Q(r)$ *is also a semi-circular relation.*

**Proof Sketch:**

Note first that closure under the standard, non-Euclidean, operators is immediate, as is closure under the Euclidean operators $\mathcal{E}_{\text{BDR}}$, $\mathcal{E}_{\text{SIDES}}$, $\mathcal{E}_{\text{CORNERS}}$, $\mathcal{E}_{i\cup j}$, $\mathcal{E}_{i\cap j}$, and $\mathcal{E}_{i-j}$. The proof will focus therefore on the remaining operators, $\mathcal{E}_{\text{RAY}}$ and $\mathcal{E}_{\text{CIRC}}$. We can show:

**Lemma 2.** *If the boundary of a set $r$ is semi-circular, so is $r$.* ∎

We now observe that RAY and CIRC are monotone, i.e., for example, $\text{RAY}(r_1, r_2 \cup r_3) = \text{RAY}(r_1, r_2) \cup \text{RAY}(r_1, r_3)$. We shall make frequent use of this fact: to start with, we may therefore assume that all input regions are connected, using an an induction argument together with monotonicity.

Lemma 2 shows that we need only show that the boundary of the *output* of an operation is semi-circular. We would like to be able to consider only the boundary of the *input* as well, using an identity such as $\text{BDR}(\text{RAY}(r, s)) = \text{RAY}(\text{BDR}(r), \text{BDR}(s))$. Unfortunately, this does not hold in general (since the RAY operation will likely produce regions), but we can show:

**Lemma 3.** *Let $r$ and $s$ be connected, non-empty, semi-circular sets. Then*

$$\text{BDR}(\text{RAY}(r, s)) = \text{BDR}(r \cup s \cup \text{RAY}(\text{BDR}(r), \text{BDR}(s))) \ .$$

This is sufficient to show that if $r$, $s$ and $\text{RAY}(\text{BDR}(r), \text{BDR}(s))$ are semi-circular sets, so is $\text{RAY}(r, s)$. To prove that $\text{RAY}(r, s)$ is always semi-circular, whenever $r$ and $s$ are, it therefore suffices to use a case analysis on $r$ and $s$ and then use the monotonicity of RAY. Several cases are illustrated in the following figures.

1. $r$: point; $s$: line segment without endpoints. $\text{RAY}(r, s)$ is the (open) region in the left side of Figure 2.
2. $r$: point; $s$: ray without endpoint. See the right side of Figure 2.

**Fig. 2.** Cases 1 and 2

3. $r$: point; $s$: circle. If $r$ is inside $s$, $\text{RAY}(r, s)$ is $\mathbb{D}^2$; otherwise, $\text{RAY}(r, s)$ is similar to the left side of Figure 3.
4. $s$: line segment. See the left side of Figure 4.
5. For $r$: circle segment, and $s$: point. See the right side of Figure 4.

This completes the proof that $\text{RAY}(r, s)$ is semi-circular. We now sketch the proof for CIRC. Let $r$, $s$ and $t$ be semi-circular sets. We show that

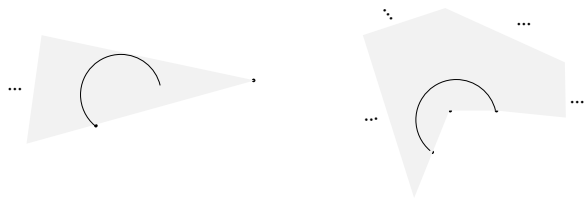

**Fig. 3.** Case 3

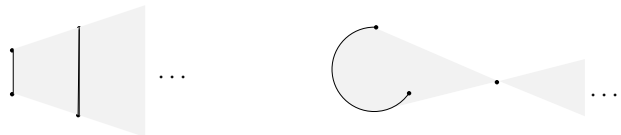

**Fig. 4.** Cases 4 and 5

$\textsc{circ}(r, s, t)$ is semi-circular. By monotonicity, we can assume that $r$, $s$, and $t$, are all connected. For the same reason, we consider the interior of $r$ and $r \cap \textsc{bdr}(r)$ separately; in fact we consider each side of the latter separately. One further assumption that we make is that the $\textsc{bdr}(r)$ is connected. Let

$$\mathcal{R} = \{d(q_s, q_t) \mid q_s \in s, q_t \in t\}$$

Then $\textsc{circ}(r, s, t)$ is the union of all circles with center in $r$ and radius equal to a number in $\mathcal{R} \subseteq \mathbb{D}^+$.[1] We claim that $\mathcal{R}$ is actually an interval. Assume that $n$, $n' \in \mathcal{R}$ and $n < n'' < n'$. Then there are points $p_s$, $q_s$ in $s$, and $p_t$, $q_t$ in $t$, such that $d(p_s, p_t) = n$ and $d(q_s, q_t) = n'$. Since $s$ and $t$ are connected, there are paths connecting $p_s$ to $q_s$, and $p_t$ to $q_t$, that are contained in $s$ and $t$, respectively. It immediately follows that there are points $p$ and $q$ on these paths with $d(p, q) = n''$.

By monotonicity, the following cases for $\mathcal{R}$ have to be considered: $[a]$, $(a, a')$, $(a, \infty)$. Most of these are proven by induction from base cases similar to rays. The main exception is the case $\mathcal{R} = (a, \infty)$. Here we need the set of points that are of distance more than $> a$ from *some* point in $r$. The idea here is to construct the complement instead, i.e., the set of all points $p$ that are of distance $\leq a$ from *every* point of $r$.

First observe that if $r$ is unbounded the result must be empty. We then proceed in two steps: (1) show that circular sides can be replaced by straight edges, without changing the result, and then (2) showing that the result is semi-circular, when $r$ is bounded semi-*linear* set.

For the first step there are two cases, depending on whether the arc is "convex" or "concave." We show here the proof works in the case of concave

[1] $\mathbb{D}^+$ is the set of numbers in $\mathbb{D}$ that are greater or equal to 0.

arc. Let $r'$ be result of replacing a concave arc in $r$ by a straight edge, and let $q_1$ and $q_2$ be the endpoints of this arc. Assume that the arc is at most half a circle (one can add an extra vertex to assure this). Since $r'$ contains $r$, it follows that if $p$ is at distance $\leq a$ from every point in $r'$, that it is at distance $\leq a$ from every point of $r$. For the converse, assume that $p$ is of distance $> a$ from some point $q$ in $r' - r$. If the line from $p$ to $q$, when extended, intersects $r$, then there must be a point in $r$ of distance $> a$ from $p$. Otherwise, it follows, using standard geometric techniques, that either $d(p, q_1)$ or $d(p, q_2)$ is greater or equal to $d(p, q)$, which is greater than $a$, by assumption.

For (2), let $r$ be semi-linear and bounded. Then it can be shown that a point $p$ is at distance $\leq a$ from every point in $r$ iff it is at distance $\leq a$ from every vertex of $r$. The latter set is the intersection of the circles with centers at the vertices of $r$ and radii $a$, hence semi-circular. ∎

## 5 Discussion and Future Work

In this paper, we have proposed a language for spatial databases defined by line segments and circles, motivated by Euclidean geometry. One natural question is how does this language relate to that of [KKPV97]? In [KS98], we show that this language in fact captures a natural fragment of $\Phi_{\mathsf{circ}}$, and that this fragment captures all of FO+lin; it would be interesting to know more about the relationship of EuAlg to traditional constraint languages, as well studying its complexity. As the Euclidean query languages can be seen as a safe restriction of other constraint languages, it would be of interest to see how they relate to the safe languages of [BL98].

Another interesting question concerns the choice of an encoding for circles. While other representations (by 3 points, or center and point on circle) may also seem reasonable approaches, and are in fact equivalent in the framework of [KKPV97], in our approach the representation seems to be critical. If we were to define $\mathcal{E}_{\text{CIRC}(i,j)}$ to be the union of all circles that have a center in region $i$ and go through a point in region $j$, the resulting language is not closed: if column 1 contains a circle, and column 2 a point, column 3 will contain a limaçon, which is known not to be semi-circular (see [Due25] for the original definition and [Loc61] for a reduction of the construction of this curve to trisection of an angle). An alternative definition, using 3 points to define a circle, appears on the the other hand to be too weak.

In spite of providing distance functions with the Euclidean construction based query languages, developing appropriate query languages for fixpoint queries remain as an interesting issue. It is unclear how EuAlg can be extended to capture fixpoint queries, as it was done for FO+lin [GK97] and for topological queries [SV98].

Finally, while the restriction to Euclidean geometry is motivated by the importance of the distance function in many spatial applications, it remains

a natural question to ask whether the current approach can be adapted to more general objects. The most obvious such extension would be to allow ellipses as well as circles, and to use as a generalization of the circle-drawing primitive the construction ellipses with radii from a given set of intervals, and foci taken from 2 given objects. Unfortunately, this breaks down even when we consider a single radius and foci on a single circle: as shown in [KS98], the language we get is not closed. It is an open question whether other approaches would be more successful.

## Acknowledgments

The authors wish to thank Jan van den Bussche and Leonid Libkin for their comments. Work by Jianwen Su is supported in part by NSF grants IRI-9411330, IRI-9700370, and IIS-9817432, and part of his work was done while visiting Bell Labs.

## References

[AB95] S. Abiteboul and C. Beeri. The power of languages for the manipulation of complex values. *VLDB Journal*, 4(4):727–794, October 1995.

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

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

[BL98] M. Benedikt and L. Libkin. Safe constraint queries. *Proc. ACM Symp. on PODS, 1998*

[BBC97] A. Belussi, E. Bertino, and B. Catania. Manipulating spatial data in constraint databases. In M. J. Egenhofer and J. R. Herring, editors, *Intl. Conf. on Advances in Spatial Databases (SSD'97)*, pages 115–141. Springer Verlag, LNCS 1262, 1997.

[Col75] G. E. Collins. Quantifier elimination for real closed fields by cylindric decompositions. In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, volume 35 of *Lecture Notes in Computer Science*, pages 134–83. Springer-Verlag, 1975.

[Due25] A. Dürer. Underweysung der Messung. Nürnberg, 1525

[GK97] S. Grumbach and G. Kuper. Tractable recursion over geometric data. In *International Conference on Constraint Programming*, 1997.

[GRS98] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In *Proc. ACM SIGMOD*, 1998.

[GS95] S. Grumbach and J. Su. Dense order constraint databases. In *Proc. ACM PODS*, 1995.

[GS97] S. Grumbach and J. Su. Finitely representable databases. *Journal of Computer and System Sciences*, 55(2):273–298, 1997.

[KKPV97] B. Kuijpers, G. Kuper, J. Paredaens and L. Vandeurzen. First Order Languages Expressing Constructible Spatial Database Queries *Journal of Computer and System Sciences*, to appear. Preliminary version appeared as J. Paredaens, B. Kuijpers, G. Kuper and L. Vandeurzen. Euclid, Tarski, and Engeler Encompassed. *Proceedings of DBPL'97*, LNCS 1369.

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

[KRSS98] G. Kuper, S. Ramaswamy, K. Shim, and J. Su. A constraint-based spatial extension to SQL. In *Proc. of ACM Symp. on GIS*, 1998.

[KS98] G. Kuper and J. Su Representation Independence and Effective Syntax of Euclidean based Constraint Query Languages. Bell Labs Technical Report 981116-13, 1998.

[Loc61] E. H. Lockwood. A Book of Curves. *Cambridge University Press*, 1961.

[PVV98] J. Paredaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over the reals. *SIAM Journal on Computing*, 27(6):1747–1763, 1998.

[SV98] L. Segoufin and V. Vianu Querying Spatial Databases via Topological Invariants *Proc. ACM Symp. on PODS*, 89–98, 1998.

[Tra50] B. A. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite models. *Doklady Akademii Nauk SSR*, 70:569–572, 1950.

[Vau60] R. L. Vaught. Sentences true in all constructive models. *Journal of Symbolic Logic*, 25(1):39–53, March 1960.

[VGV98] L. Vandeurzen, M. Gyssens, and D. Van Gucht. An expressive language for linear spatial database queries. *Proc. ACM Symp. on PODS*, 109–118, 1998.