Logical Aspects of Spatial Databases

Logical Aspects of Spatial Databases

Bart Kuijpers Jan Van den Bussche

1 Spatial data and first-order logic

In most general terms, a *spatial dataset* is any set $S \subset \mathbb{R}^n$ for some n. Equivalently, we can view such a set as an n-ary relation S over $\mathbb{R}$ (using Cartesian coordinates). Viewing $\mathbb{R}$ as a structure $\mathbb{R} = (\mathbb{R}, 0, 1, +, \cdot, <)$ over the language of ordered fields, we can then use first-order logic to express properties of spatial datasets.

For example, the sentence

$$\exists a \exists b \forall x \forall y (S(x, y) \rightarrow y = a \cdot x + b)$$

expresses that $S \subset \mathbb{R}^2$ lies on a straight line.

Since the structure on $\mathbb{R}$ in this paper remains the same, we abbreviate $(\mathbb{R}, S) \models \phi$ as $S \models \phi$.

2 Capturing first-order geometric properties

According to Felix Klein's Erlangen Programm, a geometric theory can be characterized by the group of transformations that preserve the fundamental geometric properties of the theory. Some examples:

geometry	group of transformations
Euclidean	similarity
affine	affinity
topology	continuous

Fix such a group G of transformations of $\mathbb{R}^n$, and consider some property ϕ of datasets in $\mathbb{R}^n$. We naturally define ϕ to be *G-geometric* if it is invariant under G, or formally:

$$\forall S \, \forall g \in G : S \models \phi \Leftrightarrow g(S) \models \phi$$

Let us see some examples:

- "S lies on a circle" is Euclidean, but not affine.
- "S lies on a straight line" is affine, but not topological.
- "S has dimension two" is topological.

A general question, for any fixed G, is: *What are the G-geometric properties expressible in first-order logic?* Can we enumerate or characterise them in some effective way?

This can be easily done when G is *first-order parameterisable*. By this we mean that there exists an injection $p : G \to \mathbb{R}^\ell$ for some ℓ such that the set

$$\{(p(g), \bar{x}, \bar{y}) \mid g \in G \text{ and } \bar{y} = g(\bar{x})\}$$

is first-order definable in $\mathbb{R}$.

For example, the affinities in $\mathbb{R}^2$ are first-order parameterised. Indeed, each affinity g corresponds to some 6-tuple $p(g) = (a, b, c, d, e, f)$ with

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0$$

and we have $(y_1, y_2) = g(x_1, x_2)$ iff

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

so that is all first-order definable.

The following theorem [12] provides an effective characterisation of the G-geometric first-order properties, in case G is first-order parameterisable (by the injection p):

Theorem 1. *A property Π of sets $S \subseteq \mathbb{R}^n$, for some fixed n, is first-order expressible and G-geometric if and only if $\Pi(S)$ can be expressed by a first-order sentence of the form*

$$\phi \wedge \forall p(g) \in p(G)[\phi(S) \leftrightarrow \phi(g(S))]$$

with ϕ an arbitrary sentence over $(\mathbb{R}, S)$.

Note that the first-order parameterisability of G guarantees that the special form of sentence in the above theorem is indeed a first-order sentence.

3 First-order topological properties of plain sets

When doing topology, we are only interested in properties that are invariant under "continuous transformations". More precisely, in this paper we define a property of sets in $\mathbb{R}^n$ to be *topological* if it is invariant under all *isotopies* of $\mathbb{R}^n$.

Clearly, the isotopies are not first-order parameterisable, so the easy technique from the previous section does not apply.

We are still able to capture the first-order topological properties, provided we restrict our setting to the following:

1. We work in $\mathbb{R}^2$ only, i.e., sets in the real plane.

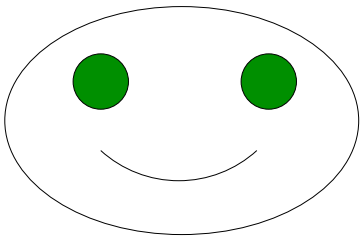


Figure 1: This set in the plane is closed, and is semi-algebraic, being defined by the real formula $x^2/25+y^2/16 = 1 \vee x^2+4x+y^2-2y \leqslant -4 \vee x^2-4x+y^2-2y \leqslant -4 \vee (x^2+y^2-2y = 8 \wedge y \leqslant -1)$.

2. We consider only *semi-algebraic* sets: sets that are themselves definable in $\mathbb{R}$.

3. Moreover, we consider only *closed* sets (in the standard topological sense).

Let us call such sets "plain". Figure 1 shows an example of a plain set [2].

Let us see some examples of topological properties of plain sets, expressible in first-order logic:

- "The dimension is 0 (or 1, or 2)".
- "There is a point where three lines intersect".
- "There is a point where two 2-dimensional regions touch".

The following topological properties of plain sets are *not* expressible in first-order logic:

- "There is a point where an even number of lines intersect".
- "The number of points where two 2-dimensional regions touch is even".
- "The set is topologically connected".

So, the question is, what are the first-order expressible topological properties of plain sets? In order to formulate our answer to this question, we use the notion of a *cone* [3].

Around any point on the boundary of any plain set, we always see a circular list of lines (L's) and regions (R's): this list is called the *cone* of the point. An illustration is given in Figure 2.

In any plain set, there may be infinitely many boundary points with cone (LL) (these are the points that lie on a line), or with cone (R) (these are the points on the boundary of a region). With the exception of these, however, there are only finitely many boundary points; this is because semi-algebraic sets have a very simple topology [2]. In particular, there are only finitely many points with a cone different from (LL) and (R); these points are called the singular

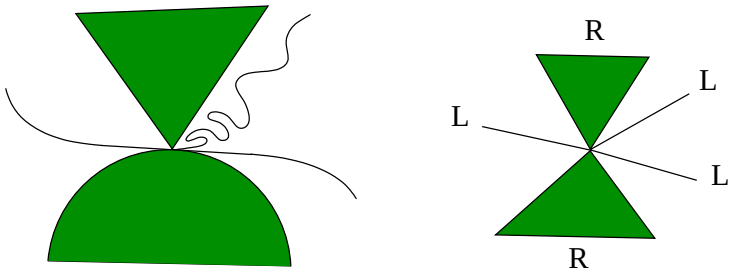


Figure 2: The cone of the central point of the figure is $(LLRLR)$.

points of the set. It can be argued [1] that, without loss of generality, we can focus on the singular points, and we will do so from now on.

We are now ready to introduce a propositional logic, called *Cone Logic* or CL for short, designed to express properties of plain sets.

- Atomic formulas are of the form

  $$|e| \geqslant n$$

  with n some natural number and e a star-free regular expression over $\Sigma = \{L, R\}$.

  The meaning of such a formula is that there are at least n singular points whose cone satisfies e. (For background on star-free regular expressions and their connection to first-order definability on strings, see McNaughton and Papert [17] and Thomas [21].)

- A CL-sentence is a boolean combination of atomic formulas.

Let us see some examples of properties expressed by CL formulas:

- "The dimension is 0":

  $$|L\Sigma^*| = 0 \wedge |R\Sigma^*| = 0$$

- "There is a point where three lines intersect":

  $$|LLLLLL| \geqslant 1.$$

- "There is a point where two regions touch":

  $$|RR| \geqslant 1$$

We can now state the following theorem [1]:

Theorem 2. *The first-order topological properties of plain sets are precisely those expressible in CL.*

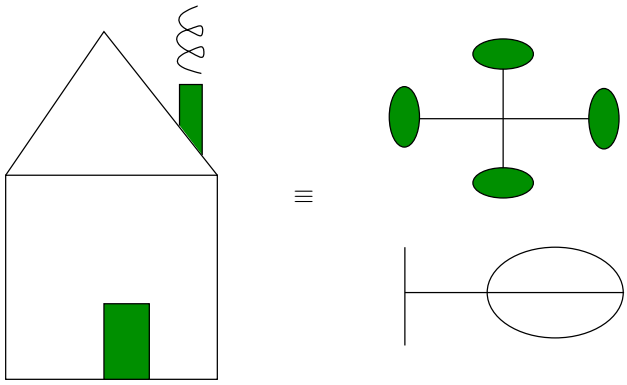


Figure 3: These two plain sets are topologically elementary equivalent, according to Theorem 3.

This theorem can be proven in six steps:

1. Topological elementary equivalence
2. Flower datasets
3. Finite structures over the reals, collapse theorems
4. Coding flower datasets by finite structures
5. Translating sentences about datasets into sentences about codes
6. Invariance arguments over codes

Let us go into these six steps in some more detail. (For the full proof, we refer to the paper [1].)

Topological elementary equivalence For plain sets A and B, write $A \equiv B$ if A and B are indistinguishable by topological first-order sentences.

We have the following theorem [14], which plays a crucial role in the proof of Theorem 2:

Theorem 3. *$A \equiv B$ if and only if A and B have precisely the same cones, with the same multiplicities.*

An illustration of this theorem is given in Figure 3.

For the full proof of this theorem we refer to the paper [14], but we give an idea of the proof here. The proof is based on a transformation of plain sets into a normal form called *flower normal form*. An illustration of this transformation is given in Figure 4.

This transformation proceeds by the use of transformation rules, such as the "cut and paste" rule illustrated in Figure 5.

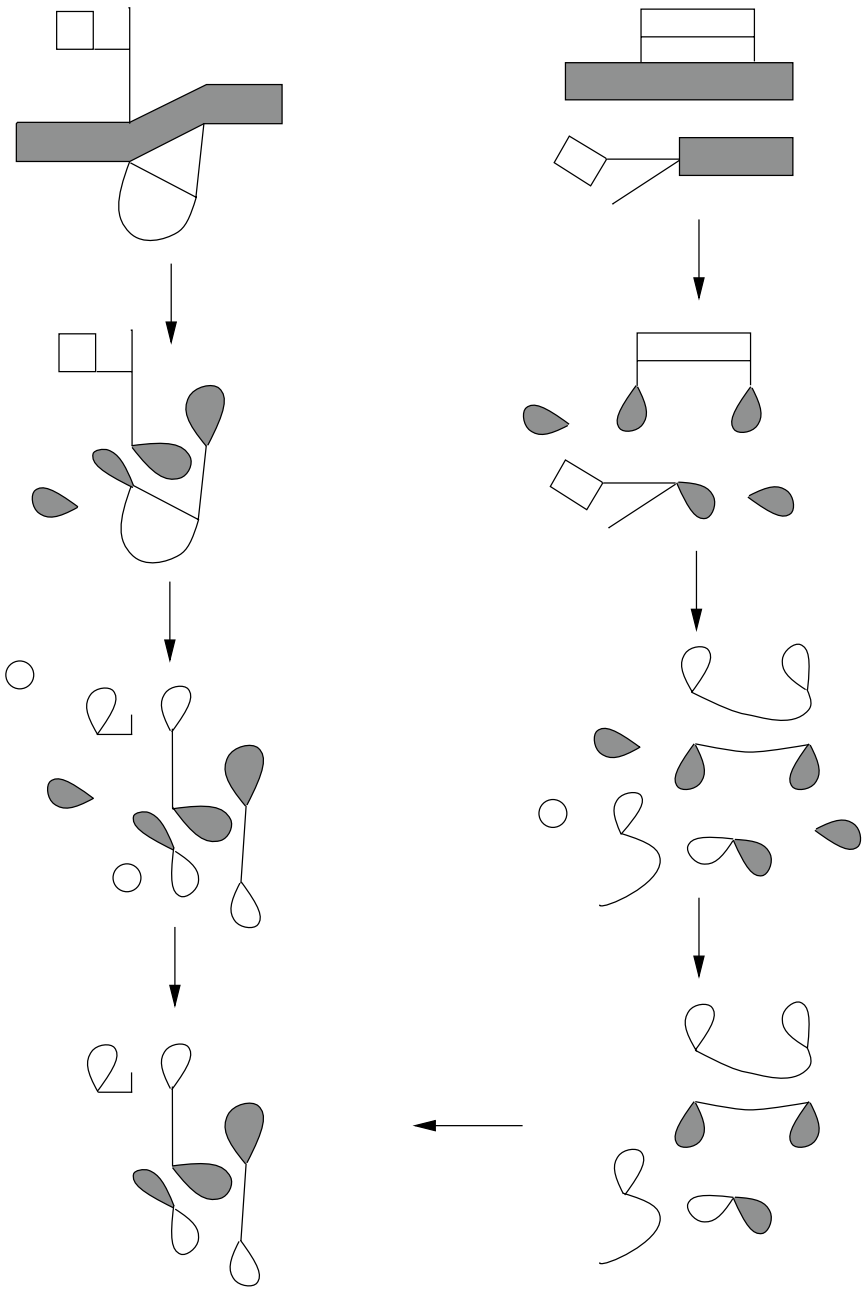


Figure 4: Two plain sets transformed into one and the same flower dataset.

Figure 5: Cut and paste.

One can show that the application of a transformation rule is indistinguishable by topological first-order sentences, using a *reduction* to first-order inexpressibility of queries on *finite structures over the reals.*

Finite structures over the reals These are structures of the form $(\mathbb{R}, R_1, \ldots, R_k)$ with R_i finite relations. An example of a query to such structures is *Majority:* given finite unary relations R_1 and R_2, is $\#R_1 \geqslant \#R_2$?

Let us illustrate the above-mentioned reduction for the cut-and-paste transformation. This reduction is done by writing a first-order formula $\psi(x, y)$ such that for each finite structure $D = (\mathbb{R}, R_1, R_2)$:

- $\psi(D)$ is homeomorphic to the left-hand side of the cut-and-paste transformation if $\#R_1 \geqslant \#R_2$ in D;
- $\psi(D)$ is homeomorphic to the right-hand side of the cut-and-paste transformation if $\#R_1 < \#R_2$ in D.

We do not give this formula ψ, but illustrate it in Figure 6. This reduction idea is due to Grumbach and Su [11].

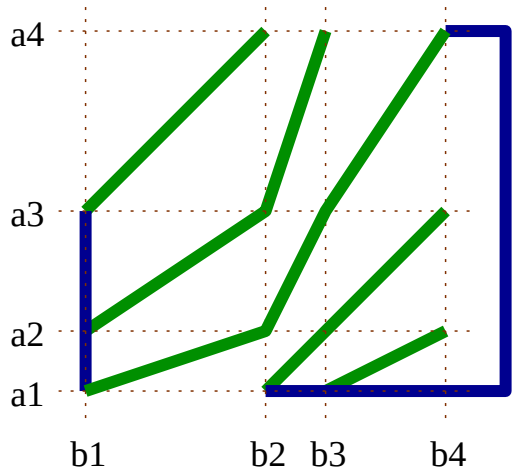
As a consequence, if the cut-and-paste transformation would be distinguishable by a topological first-order sentence, then the Majority query would be first-order expressible on finite structure over the reals. We can show the latter to be false, however, using the so-called *collapse theorems* from "Embedded Finite Model Theory" [9, Chapter 5].

Theorem 4 (Natural–active collapse)**.** *Every first-order query on finite structures over the reals is already expressible by a sentence in which all quantifiers are relativised to the finite relations.*

Theorem 5 (Generic collapse:)**.** *Every first-order query on finite structures over the reals (in the language* $(0, 1, +, \cdot, <, R_1, \ldots, R_k)$*) that is* order-generic *(invariant under all monotone permutations of* $\mathbb{R}$*) is already expressible by a sentence in the language* $(<, R_1, \ldots, R_k)$*.*

So, order-generic first-order sentences view finite structures over the reals just as abstract, ordered, finite structures. Note that the reduction ψ from above is order-generic. And the Majority query on abstract, ordered, finite structures is indeed not first-order expressible, as can be shown using standard arguments from finite model theory [4, 16].

$R_1 = \{a_1, a_2, a_3, a_4\}$, $R_2 = \{b_1, b_2, b_3, b_4\}$:



$R_1 = \{a_1, a_2, a_3, a_4\}$, $R_2 = \{b_1, b_2, b_3, b_4, b_5\}$:

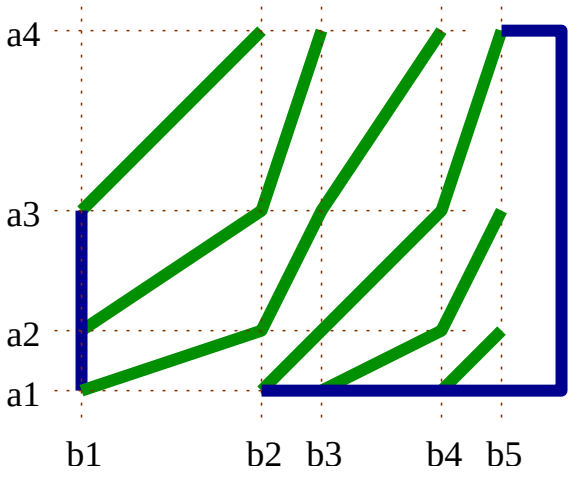


Figure 6: Reducing Majority to distinguishing the cut-and-paste transformation.

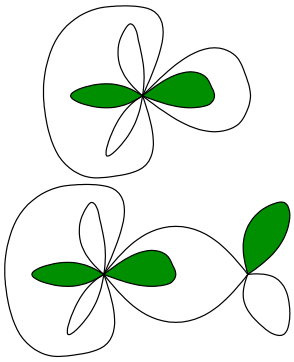


Figure 7: A single flower and a paired flower.

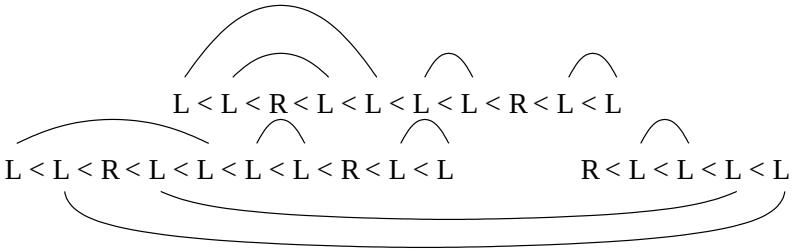


Figure 8: Cycle codings of the flowers of Figure 7.

Flower datasets Using the transformation rules, we can transform any plain dataset into a normal form (as far as topological first-order properties are concerned). This normal form is that of a disjoint union of single or paired *flowers*. A single flower has a single singular point, around which there are one-dimensional or two-dimensional "petals" (the one-dimensional petals can loop over other petals). In a paired flower, two single flowers are paired by an even number of lines that can cross over. An illustration is in Figure 7.

A flower datasets can be represent by an abstract finite structure called a *code:* this is a disjoint union of single or paired *cycles*. A single cycle is a word structure over the alphabet $\{L, R\}$ equipped with a planar matching. In a paired cycle, there are two words, and there is again a planar matching now on all the L's, so the matching can again cross over. An illustration is in Figure 8.

Translation argument By a translation argument, we can now reduce the proof of Theorem 2 to an invariance question about first-order logic over codes. This translation argument is based on the following [1]:

Lemma 1 (Drawing Lemma)**.** *We can write an FO-formula $\delta(x, y)$ such that for any code C embedded in the reals, $\delta(C)$ is a flower dataset that is a drawing*

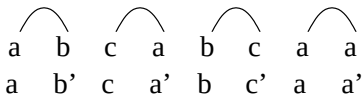


Figure 9: Chain matching represented as a word with alternating markers.

of C.

The presence of the planar matching in codes was obviously meant to facilitate such a drawing lemma.

The lemma allows us to translate a topological sentence ϕ about flower datasets into a sentence $\psi := \phi \circ \delta$ about codes, called an *implementation* of ϕ. Using the collapse theorems, we may assume that ψ sees only an ordered version of the abstract code. This ordering $<$ is not the $\prec$ of the word structures, but without loss of generality, we can actually assume that $<$ does agree with $\prec$, so all $<$ does is shuffle the separate cycles in some order. Note, however, that ψ is *invariant* under the way this shuffling is done!

Now such $<$-invariant first-order sentences on ordered codes can be seen to be already expressible by plain first-order sentences on codes. So, we are closing in on our goal, as first-order logic on codes comes already quite close to Cone Logic. The "only" difference is that codes still contain a planar matching, which Cone Logic lacks. Crucially, however, a rather technical argument shows that ψ is *also* invariant under the particular choice of planar matching in a code. We are thus faced with one final hurdle, which is removed by the following.

Planar-matching-invariant FO on word structures Our general result concerning word structures over a finite alphabet Σ, additionally equipped with a planar matching G, is the following:

Lemma 2 (Main Invariance Lemma)**.** *G-invariant first-order logic collapses to plain first-order logic on the class of word structures with a planar matching.*

The lemma is formulated for standard word structures, but can be adapted to cycles and cycle pairs.

The proof of the lemma focuses on two particular kinds of planar matchings: *chain matchings* and *parenthetical matchings*. A chain matching, as illustrated in Figure 9, can be simulated by relabeling every other position in the word with a marked letter (using a second alphabet Σ' with marked letters). A parenthetical matching, as illustrated in Figure 10, can be simulated by "folding" the word, moving to the alphabet Σ^2.

We can now translate first-order logic over words with chain matchings to first-order logic over words with alternating markers; and we can likewise translate first-order logic over words with parenthetical matchings to first-order logic over folded words. Both translations imply that set W of words accepted by a planar-matching-invariant sentence is surely regular, and can have only very

Figure 10: Parenthetical matching represented as a folded word.

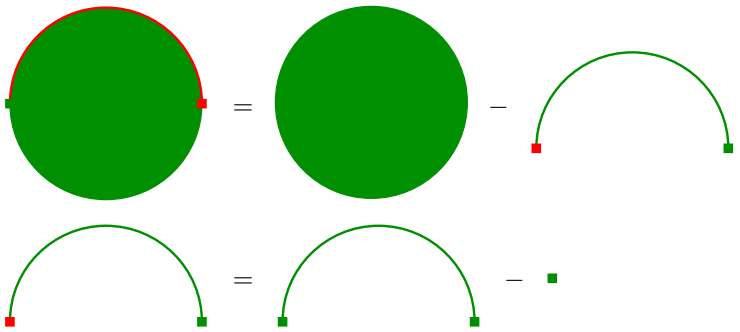


Figure 11: Every set is a boolean combination of closed sets.

limited kind of *counters* in the sense of McNaughton [17]. A final, complicated, argument then shows that $W = W' \cap (\Sigma\Sigma)^*$ with W' a counter-free regular language; since counter-free regular languages are first-order axiomatisable, the Lemma is proved.

4 Conclusion on first-order topological properties

A corollary of Theorem 2 is what can be considered a topological version, for (typically infinite) real datasets, of the earlier-mentioned collapse theorems for finite structures over the reals:

Corollary 1 (Topological Collapse)**.** *Every topological first-order property of plain sets S is already expressible by a sentence using only $<$ and S.*

This corollary follows because Cone Logic can already be expressed in first-order logic over $(\mathbb{R}, S)$ using only $<$ and S.

A natural open problem is to go beyond plain sets. What about non-closed sets? We can always decompose a set in $\mathbb{R}^n$ in $n+1$ closed sets, as illustrated in Figure 11. Hence, the more general question is, what about the first-order topological properties of ensembles of plain sets, as opposed to single plain sets? Grohe and Segoufin [10] have shown that the situation there is considerably more complex.

Another very natural open question is to move to higher dimensions: capturing the first-order topological properties of semi-algebraic sets in $\mathbb{R}^3$.

And, what about non-semialgebraic sets? Consider, for example, the property "every point in the set has cone (LL)" (i.e., the dataset consists of a number of disjoint curves either closed or going to infinity). This is first-order expressible, and it is topological over semi-algebraic sets, but it is not topological over all sets (semi-algebraic sets have a very tame topology). Can one find also an example of a topological property that is first-order expressible over semialgebraic sets, but not over all sets?

5 Point-based logics for geometric queries

First-order logic over $\mathbb{R}$ is clearly a coordinate-based logic. Cone Logic, on the other hand, is a point-based logic, as it deals directly with the (singular) points in a dataset. Can we find point-based logics for other kinds of geometric queries? The answer is affirmative, thanks to an observation made by Tarski [20] to the effect that *the geometric constructions of addition and multiplication are first-order expressible using a single ternary predicate β ("betweenness") on points.* These constructions are illustrated in Figure 12.

Let us see how we can use this to obtain a point-based logic for the first-order affine queries. In this logic, we view a dataset $S \subset \mathbb{R}^2$ as a *unary* relation over the structure $(\mathbb{R}^2, \beta)$, and we use plain first-order logic over the universe $\mathbb{R}^2$, with ternary β (betweenness) and unary S as the only predicates (as always, β is fixed and belongs to the background, while S is variable and represents the input to the query). Note how this setup differs from the coordinate-based approach, where we view S as a *binary* relation over $(\mathbb{R}, 0, 1, +, \cdot, <)$. Let us denote the point-based logic by FO(β), and let us denote the coordinate-based logic by FO($\mathbb{R}$). Both are first-order logics.

It now turns out that we can simulate FO($\mathbb{R}$) by FO(β) in the following sense. Call a triple (o, e_1, e_2) of non-collinear points, a *basis*. Then for each FO($\mathbb{R}$)-sentence ϕ there exists an FO(β)-formula $\psi(o, e_1, e_2)$ such that for every dataset S and for every basis (o, e_1, e_2):

$$S \models \psi(o, e_1, e_2) \quad \Leftrightarrow \quad \alpha(S) \models \phi$$

where α is the unique affinity that maps (o, e_1, e_2) to $((0,0),(1,0),(0,1))$.

As a corollary, we obtain [12]:

Theorem 6. *For each* FO($\mathbb{R}$)*-sentence ϕ expressing an affine geometric query there exists an equivalent* FO(β)*-sentence ψ (and vice versa).*

Adding the 4-ary equidistance predicate, we can likewise capture the first-order Euclidean queries [12].

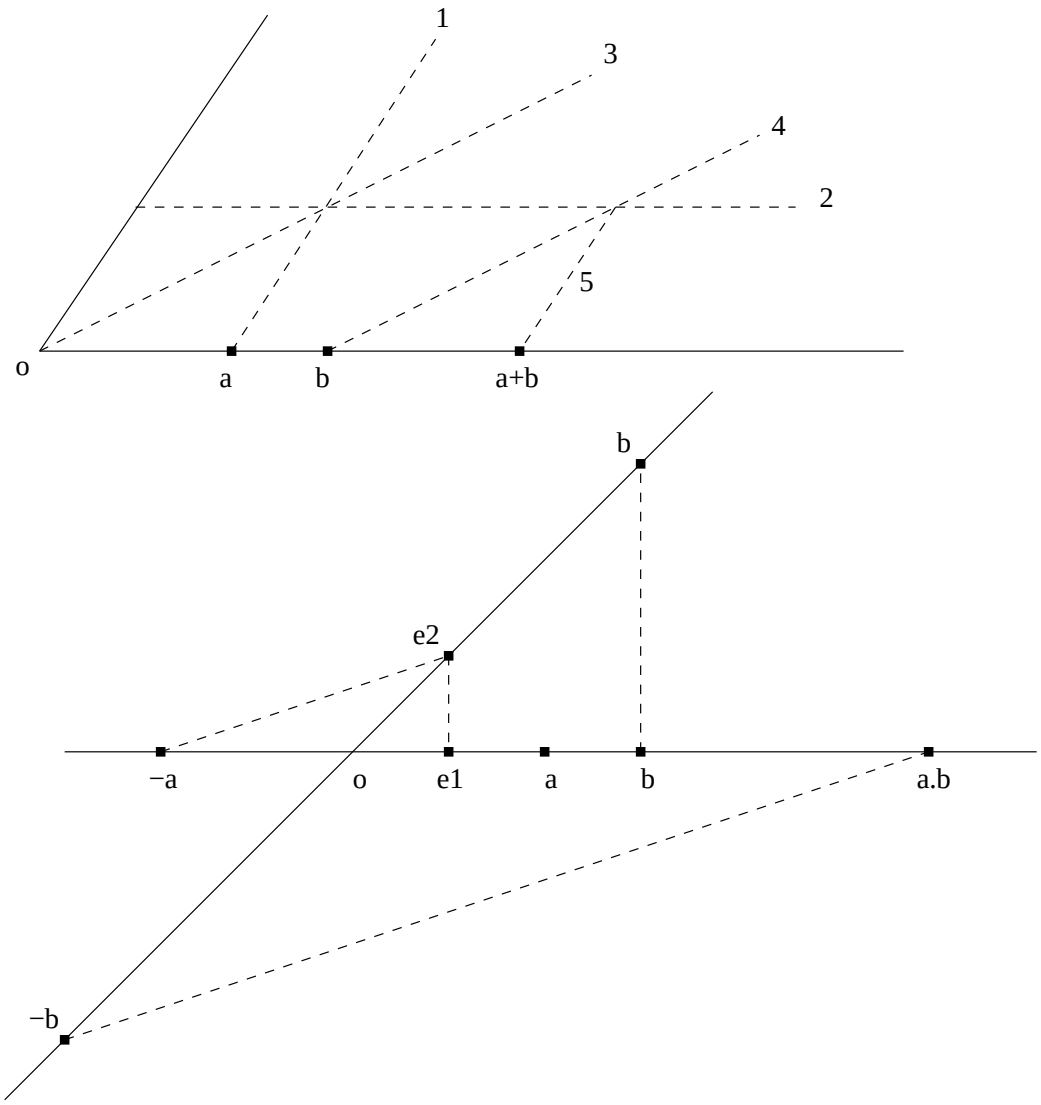


Figure 12: Geometric constructions of addition and multiplication.

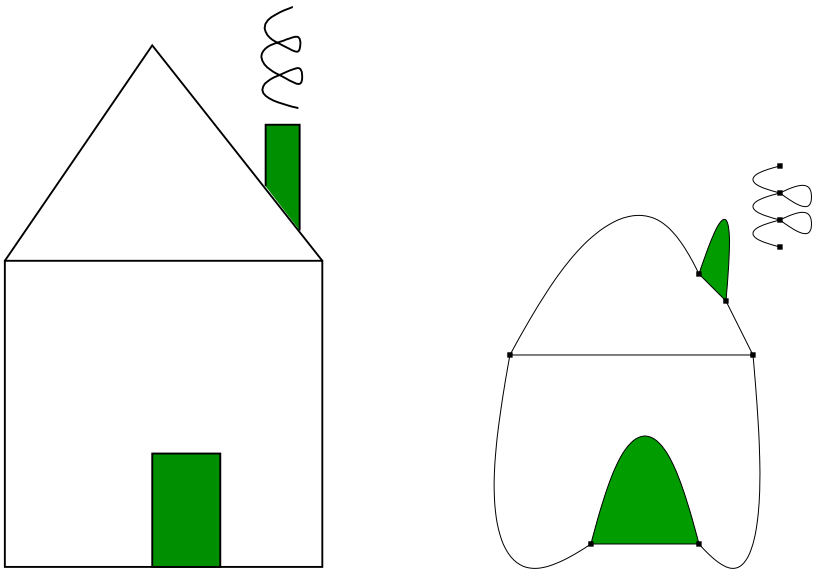


Figure 13: Plane graph data structure representing the topology of a semialgebraic set.

6 Plane graphs

To conclude this survey we must mention a very nice result by Segoufin and Vianu [19]. The topology of a semialgebraic set in the plane can be represented by a finite data structure called a plane graph. This is illustrated in Figure 13.

We have [19]:

Theorem 7. *Every topological first-order sentence about semialgebraic sets in the plane, using only $<$ and S, can be translated to a first-order sentence about the corresponding plane graphs.*

By topological collapse (see Section 4), we know that (for a single plain set at least) the restriction to only $<$ in the above statement is harmless.

7 Spatial datalog and first-order logic extend with a while-loop

Topological connectivity is a property that is important in many applications, in particular in geographical information systems (GIS). As we remarked in Section 3, topological connectivity is not expressible in first-order logic and several, more expressive extensions of first-order logic over the reals have been proposed that do allow the expression of topological connectivity of spatial datasets. In

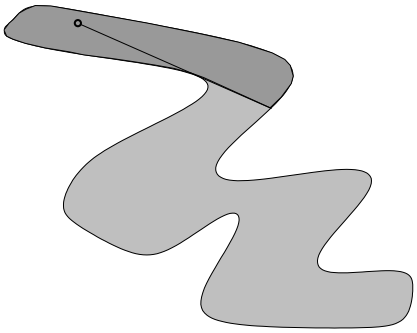


Figure 14: Linear path connectivity of a spatial dataset in the plane.

this section, we briefly discuss two such extensions: spatial Datalog and first-order logic extended with a while-loop. In the next section, we will discuss in more detail extensions of first-order logic with different types of transitive-closure operators.

Spatial Datalog Essentially, the query language *spatial Datalog* is Datalog extended with polynomial inequalities in the body of rules, with the understanding that the underlying domain is $\mathbb{R}$; that the only extended database predicate is S (the input spatial dataset); and that relations can be infinite [15].

The following spatial Datalog program expresses linear-path connectivity of a two-dimensional spatial dataset. In the relation $Obstr(x, y, x', y')$ couples of points of S are stored that cannot be connected by a straight line segment that is entirely in S. Couples of points that are not obstructed are collected in the *Path* relation (see Figure 14) and next the transitive closure of *Path* is computed. Only if all pairs of points in S end up in *Path*, S is connected.

$$
\begin{array}{rcl}
Obstr(x,y,x',y') & \longleftarrow & \neg S(\bar{x},\bar{y}), S(x,y), S(x',y'), \\
 & & \bar{x} = a_1 t + b_1, \\
 & & \bar{y} = a_2 t + b_2, 0 \leqslant t, t \leqslant 1, \\
 & & b_1 = x, b_2 = y, \\
 & & a_1 + b_1 = x', \\
 & & a_2 + b_2 = y' \\
Path(x,y,x',y') & \longleftarrow & \neg Obstr(x,y,x',y') \\
Path(x,y,x',y') & \longleftarrow & Path(x,y,x'',y''), \\
 & & Path(x'',y'',x',y') \\
Disconnected & \longleftarrow & S(x,y),\ S(x',y'), \\
 & & \neg Path(x,y,x',y') \\
Connected & \longleftarrow & \neg Disconnected.
\end{array}
$$

To show that this spatial Datalog program correctly tests topological *connectivity* on a class $\mathcal{C}$ of spatial datasets, we have to show that for any set S

in $\mathcal{C}$, that 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 (soundness); and that the number of line segments needed to connect any such pair of points in S is bounded (termination). 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.

We have the following result [15].

Theorem 8. *The above spatial Datalog program correctly tests connectivity of linear spatial datasets (i.e., spatial datasets that can be described using addition only) .*

In fact, this program correctly tests topological connectivity for a wider class of spatial datasets [15], but not for arbitrary semi-algebraic figures in $\mathbb{R}^2$. For example, when the input set is the area between the parabola given by $y = x^2$ and $y = 2x^2$, then soundness is satisfied for all points except the origin. Even when the origin is left out, termination is violated since there is no uniform bound on the number of line segments needed to connect two points. The closer one point gets to the origin, the more segments are needed.

First-order logic extended with a while-loop Another extension is first-order logic with a while-loop. Basically, in this language first-order definable relations can be created using the input spatial dataset and previously created relations. Also, a while-loop with a first-order expressible stop-condition is allowed. These two constructs are illustrated in the following program to test linear-path connectivity.

$Seg := \{(x, y, x', y') \mid \forall\lambda(0 \leqslant \lambda \leqslant 1 \wedge \forall u \forall v((u, v) = \lambda(x, y) + (1 - \lambda)(x', y') \rightarrow S(u, v)))\};$
$Path_1 := Seg;$
$Path_2 := \{(x, y, x', y') \mid \exists u \exists v(Path_1(x, y, u, v) \wedge Seg(u, v, x', y'))\};$
while $Path_1 \neq Path_2$
do
$Path_1 := Path_2;$
$Path_2 := \{(x, y, x', y') \mid \exists u \exists v(Path_1(x, y, u, v) \wedge Seg(u, v, x', y'))\};$
od
$R_{out} := \{() \mid \forall x \forall y \forall x' \forall y'((S(x, y) \wedge S(x', y')) \leftrightarrow Path_2(x, y, x'x, y')))\};$

In the Seg relation, couples of points are collected that can be connected by a line segment that is completely in S. Next, in the while-loop, the transitive closure of this relation is computed and the output relation R_{out} reflects whether all pairs of points of S are in this transitive closure.

This example shows that topological connectivity of linear spatial datasets can be expressed in this language, but in fact we have the following more powerful result [12].

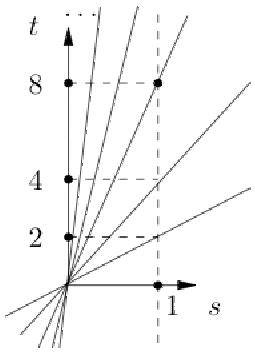


Figure 15: A non-terminating transitive-closure computation.

Theorem 9. *First-order logic extended with a while-loop is a computationally complete language on spatial (semi-algebraic) datasets.*

8 First-order logic extended with transitive-closure operators

In the previous section, we have seen that, at least for linear datasets, the expression of the transitive closure (TC) suffices to express connectivity. In this section, we describe the extension of first-order logic with various transitive-closure operators more extensively.

We cannot add TC with it standard mathematical semantics, indeed $\mathrm{TC}(\{(x,y) \mid y = 2x\})$ is not a semi-algebraic set. We look at the TC-operator as a programming construct with a purely operational semantics and therefore $\mathrm{TC}(\{(x,y) \mid y = 2x\})$ is regarded as a non-terminating computation.

Transitive-closure logic More precisely, first-order logic over the reals is extended with expressions of the form

$$[\mathrm{TC}_{\vec{x};\vec{y}}\, \psi(\vec{x},\vec{y})](\vec{s},\vec{t})$$

where $\vec{x}$, $\vec{y}$ are k-tuples of real variables. The evaluation on a input dataset A is then obtained as follows. We set $X_1 := \psi(A)$, and $X_{i+1} := X_i \cup \{(\vec{x},\vec{y}) \in \mathbb{R}^{2k} \mid (\exists \vec{z})\,(X_i(\vec{x},\vec{z}) \wedge X_1(\vec{z},\vec{y}))\}$, and stop the computation as soon as $X_{i+1} = X_i$. The semantics of $[\mathrm{TC}_{\vec{x};\vec{y}}\, \psi(\vec{x},\vec{y})](\vec{s},\vec{t})$ is then defined as the $2k$-ary relation $X_i(\vec{s},\vec{t})$.

For example, $[\mathrm{TC}_{x;y}\, S(x,y)](s,t)$ evaluated on $A = \{(x,y) \mid y = 2x\}$ gives $X_1 = \{(s,t) \mid t = 2s\}$; $X_2 = X_1 \cup \{(s,t) \mid t = 4s\} = \{(s,t) \mid t = 2s \vee t = 4s\}$; $X_3 = X_2 \cup \{(s,t) \mid t = 8s\} = \{(s,t) \mid t = 2s \vee t = 4s \vee t = 8s\}$; ... which is a non-terminating computation (illustrated in Figure 15).

On the other hand, the connectivity of linear spatial datasets in the plane can be expressed by the formula $\forall\vec{x}\forall\vec{y}(S(\vec{x}) \wedge S(\vec{y}) \rightarrow [\mathrm{TC}_{\vec{r},\vec{s}}(Seg(\vec{r},\vec{s})](\vec{x},\vec{y}))$ with $Seg = \{(\vec{r},\vec{s}) \mid (\exists\lambda)(0 \leqslant \lambda \leqslant 1 \wedge (\forall\vec{t})((\vec{t} = \lambda\cdot\vec{r} + (1-\lambda)\cdot\vec{s}) \rightarrow S(\vec{t})))\}$. On linear sets this expression gives rise to a terminating computation.

Transitive-closure logic with stop conditions A variant of the above transitive-closure logic is first-order logic with expressions of the form

$$[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y}) \mid \sigma](\vec{s},\vec{t})$$

where additionally σ is an first-order definable stop condition. The evaluation of this expression on input database A is again the computation of X_1, X_2, $X_3, \ldots$ as above but with the additional stop condition $(A, X_{i+1}) \models \sigma$. Remark that we do not allow parameters in TC-expressions.

For example, if $[\mathrm{TC}_{x;y}\,S(x,y) \mid X(1,8)](s,t)$ is evaluated on $A = \{(x,y) \mid y = 2x\}$, then $X_1 = \{(s,t) \mid t = 2s\}$; $X_2 = X_1 \cup \{(s,t) \mid t = 4s\} = \{(s,t) \mid t = 2s \vee t = 4s\}$; $X_3 = X_2 \cup \{(s,t) \mid t = 8s\} = \{(s,t) \mid t = 2s \vee t = 4s \vee t = 8s\}$; and the computation terminates because $(1,8) \in X_3$.

The above two languages were introduced in [7].

K-transitive-closure logic Another variant of transitive-closure logic was proposed by Kreutzer [13] and we call it *K-transitive-closure logic*. In K-transitive-closure logic the transitive-closure operator may be applied to parameterized sets and the evaluation of a transitive-closure expression may be controlled by the termination of particular paths in its computation rather than by the termination of the transitive closure of the complete set.

More formally, it is first-order logic over the reals extended with expressions of the form

$$[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y},\vec{u})](\vec{s},\vec{t}),$$

where $\vec{u}$ is an ℓ-tuple. The evaluation on an input dataset A is obtained in stages as follows. First, we set $X_1 := \psi(A) \wedge \bigwedge_{i\in I}(s_i = x_i)$; and then continue $X_{i+1} := X_i \cup \{(\vec{x},\vec{y},\vec{u}) \in \mathbb{R}^{2k+\ell} \mid (\exists\vec{z})\,(X_i(\vec{x},\vec{z},\vec{u}) \wedge \psi(\vec{z},\vec{y},\vec{u}))\}$; and we stop the computation as soon as $X_i = X_{i+1}$. The semantics of $[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y},\vec{u})](\vec{s},\vec{t})$, is then defined to be X_i. An example follows in Section 10.

We remark that the initial transitive-closure logic is a subset of K-transitive-closure logic.

9 Expressiveness properties of transitive-closure logics

For the transitive-closure logic with stop condition we have the following expressiveness result.

Theorem 10. *All computable queries on linear spatial datasets, definable by linear polynomial with coefficients in $\mathbb{Z}$, are expressible in the transitive-closure logic with stop conditions.*

The previous result even holds when we disallow the use of multiplication.

The proof of Theorem 10 can be sketched as follows. Let Q be a computable query on linear spatial datasets. Then we will write Q as a composition $Q_5 \circ Q_4 \circ Q_3 \circ Q_2 \circ Q_1$ of five queries that are expressible in transitive-closure logic with stop conditions. To start with, Q_1 produces on input a spatial dataset S in $\mathbb{R}^n$, a finite relation contains $(n+1)^2$-tuples, describing a triangulation of S. Since S can described by linear polynomial with coefficients in $\mathbb{Z}$, these corner points of the triangulation will be rational numbers. This encoding (and de corresponding decoding Q_5) can actually be done in first-order logic over the reals.

The queries Q_2 and Q_4 are the encoding/decoding of finite relations over the rational numbers into single natural numbers. Finally Q_3 is the query that simulates Q on the natural number encodings of spatial datasets. The existence of a formula for Q_3 is guaranteed by the following powerful lemma.

Lemma 3. *For every partial computable function $f : \mathbb{N}^k \to \mathbb{N}$ there exists a formula $\varphi_f(y)$ in transitive-closure logic with stop conditions over the schema $\mathcal{S} = \{S^{(k)}\}$, such that for any database D over $\mathcal{S}$ with $S^D = \{(n_1, ..., n_k)\}$, we have that $\varphi_f(D)$ is defined if and only if $f(n_1, \ldots, n_k)$ is defined, and in this case $\varphi_f(D) = \{f(n_1, \ldots, n_k)\}$.*

This can be shown by simulating the run of a non-deterministic p-counter machine $M_f = (Q, \delta, q_0, q_f)$ which computes f.

We remark that the above theorem is limited to linear spatial datasets and that for arbitrary it is not known whether a finite encoding of a semi-algebraic set can be expressed in transitive-closure logic with stop conditions. But for queries on arbitrary spatial datasets we have completeness if we restrict our attention to topological properties.

Theorem 11. *All computable* Boolean topological *queries on spatial datasets are expressible in transitive-closure logic with stop conditions.*

This theorem can be proven by showing that we can approximate a spatial dataset by a $\mathbb{Z}$-linear spatial database that is toplogically equivalent to it in transitive-closure logic with stop conditions and by using the first expresiveness (Theorem 10) result.

To be more precise about the approximation, we now describe how a rational ε-approximation of a spatial dataset can be expressed in transitive-closure logic with stop conditions. The task here is, given a spatial dataset S and a real number $\varepsilon > 0$, to find a $\mathbb{Z}$-linear spatial dataset that is homeomorphic to S (topological condition) and that ε-approximates S (metric condition).

It is not difficult to show that ε-approximations cannot be expressed in first-order logic over the reals [6]. We now sketch, for the case of $\mathbb{R}^2$, how a rational ε-approximation of a spatial dataset S can be expressed in transitive-closure logic with stop conditions.

First, we find all points where the boundary of S is not smooth and border points with a vertical tangent line. For these points, we compute their local cone

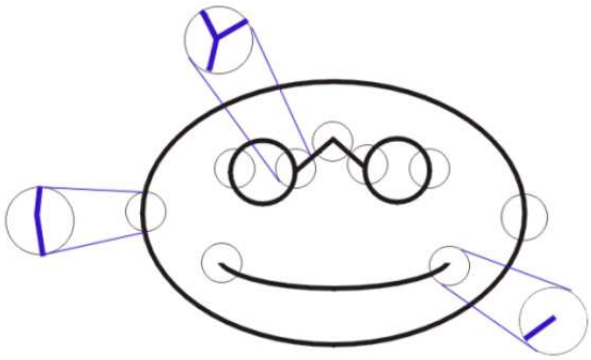


Figure 16: Local rectification around non-smooth border points and points with a vertical tangent line.

radius (conicity around points was discussed in Section 3) and within these radii we locally rectify the database. This is all expressible in first-order logic over the reals [5, 8] and illustrated in Figure 16.

Next, we consider the border of S outside the cone radii determined in the first step. What remains of the border are simple curves. We then compute the minimal cone radius r of all points on these curves in first-order logic. Let $Step$ be the relation of pairs of points (p, q) on these curves such that $d(p, q) = r$. In transitive-closure logic, we can compute $\mathrm{TC}(Step)$ and the termination of this computation is guaranteed. Once this is done, we walk over these curves starting from the endpoints and locally rectify them. This is illustrated in Figure 17. We can also do this when the curves are bordering curves of the interior.

Finally, we glue the result of the first step onto the result of the second step. The final result is a $\mathbb{Z}$-linear database homeomorphic to the original semi-algebraic set (illustrated in Figure 18).

In dimensions higher than 2, the description of rational ε-approximation in first-order logic with transitive closure with stop condition technically more complicated. It follows a recursive procedure on the dimension and boxes are used instead of spheres to describe the cones of points.

We end this section with a corollary of Theorem 11 that concerns the, for applications such as geographical information systems, important property of connectivity.

Corollary 2. *Topological connectivity of (even non-linear!) spatial datasetsis expressible in transitive-closure logic with stop conditions.*

The results in this section can be found in [6, 8].

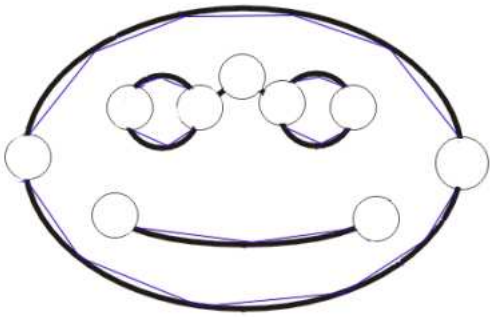


Figure 17: Rectification of the border away from non-smooth border points.

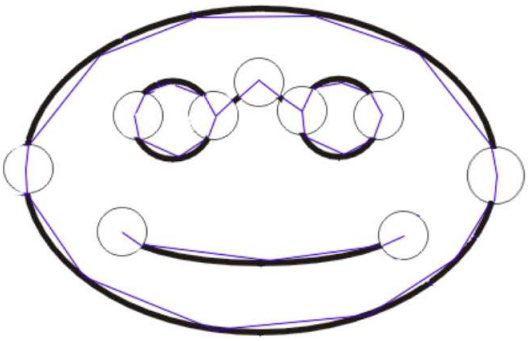


Figure 18: Glueing the result of the two steps together.

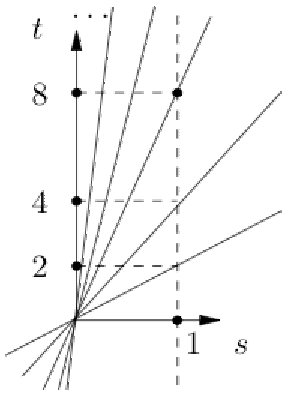


Figure 19: An example of a non-terminating evaluation of transitive closure.

10 Deciding termination of transitive-closure logic expressions

From the definition in Section 7, it is clear that the evaluation of queries expressed in transitive-closure logic with or without stop conditions may be non-terminating.

Some examples For example, if we evaluate the transitive closure, that is, the expression $[\mathrm{TC}_{x;y}\, S(x,y)](s,t)$, on the spatial dataset $A = \{(x,y) \mid y = 2x\}$, we get $X_1 = \{(s,t) \mid t = 2s\}$; $X_2 = X_1 \cup \{(s,t) \mid t = 4s\} = \{(s,t) \mid t = 2s \vee t = 4s\}$; $X_3 = X_2 \cup \{(s,t) \mid t = 8s\} = \{(s,t) \mid t = 2s \vee t = 4s \vee t = 8s\}$, that is a growing number of lines. In other words, this is a non-terminating computation (which is illustrated in Figure 19).

Even if we modify the function $y = 2x$, as shown by the thick line in Figure 20, that is, even if we bound its image, the computation of the transitive closure remains non-terminating (as illustrated in Figure 20).

But there are function graphs on which the computation of the transitive closure terminates. The function shown by the thick line in Figure 21, is an example. Here we have a terminating computation because $X_4 = X_5 = X_6 = \cdots$ (illustrated by the thinner lines in Figure 21).

Also, when we consider stop-conditions, the evaluation of the very basic transitive-closure query may lead to non-terminating computations. Obviously, adding stop-conditions may make non-terminating computations terminating, as is illustrated by the following examples which continue the above examples.

If we apply the query expressed by $[\mathrm{TC}_{x;y}\, S(x,y) \mid X(1,8)](s,t)$ to the dataset $A = \{(x,y) \mid y = 2x\}$ given in Figure 19, the computation terminates because, $(1,8) \in X_3$.

Also, when $[\mathrm{TC}_{x;y}\, S(x,y) \mid \exists x \exists y X(x,y) \wedge y = 1 \wedge 10x < 1](s,t)$ is applied to

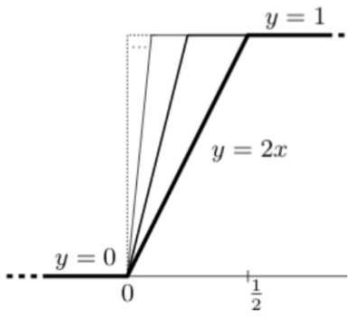


Figure 20: A second example of a non-terminating evaluation of transitive closure.

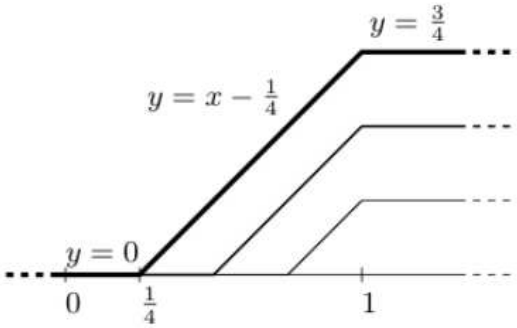


Figure 21: An example of a terminating evaluation of transitive closure.

the function graph given in Figure 20, we get a terminating computation because the fourth set we compute satisfies the stop condition, that is, $\exists x \exists y X_4(x,y) \wedge y = 1 \wedge 10x < 1$.

Obviously, also K-transitive closure logic has the problem of non-terminating evaluations, since it contains transitive-closure logic. We give an example of a terminating evaluation. Consider the evaluation of $[\mathrm{TC}_{x;y}\, S(x,y)](\frac{1}{4}, t)$ on the graph of the function given in Figure 20. This gives $X_1 = \{(\frac{1}{4}, \frac{1}{2})\}$, $X_2 = \{(\frac{1}{4}, \frac{1}{2}), (\frac{1}{2}, 1)\}$ and $X_3 = X_2$.

Deciding termination and undecidability results We can ask whether there is a procedure to decide, for given an expression φ in some transitive-closure logic and a dataset A whether φ has terminating evaluation on A?

It is fairly easy to obtain the following undecidability results.

Theorem 12. *It is undecidable whether a given formula in transitive-closure logic, that uses transitive closure on relations of at most arity 4, terminates on a given input database.*

The proof of this fact is by reduction of undecidability of nilpotency of a piecewise affine function $f : \mathbb{R}^2 \to \mathbb{R}^2$ to this problem (a function $f : \mathbb{R}^n \to \mathbb{R}^n$ is called *nilpotent* if there is a $k \geqslant 1$ such that for all $\vec{x} \in \mathbb{R}^n$: $f^k(\vec{x}) = \vec{0}$).

The following is an immediate consequence.

Corollary 3. *It is undecidable whether a given formula in K-transitive-closure logic, that uses transitive closure on relations of at most arity 4, terminates on a given input database.*

The following theorem can be proven using the undecidability of Hilbert's 10th problem.

Theorem 13. *It is undecidable whether a given formula in transitive-closure logic with stop conditions, that uses transitive closure on at most binary relations, terminates on a given input database.*

These results are complete for the languages all three types of transitive-closure logics that we have considered, apart from the cases of (K-)transitive-closure restricted to work on binary relations.

We have the following open problem: *Is it decidable whether a given formula in K-transitive-closure logic restricted to binary relations terminates on a given input database?* This problem is related to an open problem in dynamical systems theory, namely the *point-to-fixed-point problem* which asks whehther for a given algebraic number x_0 and a given piecewise affine function $f : \mathbb{R} \to \mathbb{R}$, the sequence $x_0, f(x_0), f^2(x_0), f^3(x_0), \ldots$ reaches a fixed point? This decision problem is open, even for two pieces.

We have a second open problem: *Is it decidable whether a given formula in transitive-closure logic restricted to binary relations terminates on a given*

input database? In this case, this problem is related to deciding nilpotency of functions $f : \mathbb{R}^n \to \mathbb{R}^n$. That is, deciding termination of

$$[\mathrm{TC}_{\vec{x};\vec{y}}\, S(\vec{x},\vec{y})](\vec{s},\vec{t})$$

applied to graph(f) adds up to deciding nilpotency of f (a function $f : \mathbb{R}^n \to \mathbb{R}^n$ is *nilpotent* if there exists a number $k \geqslant 1$ such that for all $\vec{x} \in \mathbb{R}^n$: $f^k(\vec{x}) = \vec{0}$). But nilpotency of (possibly discontinuous) functions $f : \mathbb{R}^n \to \mathbb{R}^n$ is not known to be undecidable for $n = 1$, whereas it is for $n > 1$.

Terminating functions We say that a set $A \subseteq \mathbb{R}^2$ has *terminating transitive closure* if the formula $[\mathrm{TC}_{x;y}\, S(x,y)](s,t)$ terminates on input A and we call a function $f : \mathbb{R} \to \mathbb{R}$ *terminating* if graph(f) has a terminating transitive closure.

The function of Figure 19 is not terminating, but the one of Figure 21 is.

We have the following, for what follows, important result.

Theorem 14. *There is a procedure that on input a continuous semi-algebraic function $f : \mathbb{R} \to \mathbb{R}$ decides whether it is terminating. Furthermore, this decision procedure is expressible in first-order logic over the reals.*

To get to this result, we need some terminology and lemma's. Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. We call $x \in \mathbb{R}$ is a *periodic point of f* if $f^d(x) = x$ for some $d \geqslant 1$ and the smallest such d is the *period of x* . Let $\mathrm{Per}(f)$ denote the set of periodic points of f.

We have the following properties.

Lemma 4. *If f is continuous and terminating, then* $\mathrm{Per}(f) = f^k(\mathbb{R})$ *(for some k) is non-empty, closed and connected. Furthermore,* $\mathrm{Per}(f) = \{x \in \mathbb{R} \mid f^2(x) = x\}$.

A result by Sharkovskiĭ's from 1964 implies that if f is continuous and terminating, then only periods $1, 2, 4, \ldots, 2^d$ can appear for some integer value $d \geqslant 1$. The last part of the lemma is more specific.

So, we can conclude the following.

Corollary 4. *If f is continuous and terminating, then f can only have periodic points with periods 1 and 2.*

We give the following crucial lemma without proof (see [7] for details).

Lemma 5. *There is an first-order sentence that expresses whether a continuous semi-algebraic function $f : \mathbb{R} \to \mathbb{R}$ is nilpotent.*

Using the above results, we obtain the following procedure to decide termination of a function $f : \mathbb{R} \to \mathbb{R}$.

Algorithm TERMINATE(input f):
Step 1. Compute the sets $C_1 = \{x \mid f(x) = x\}$ and $C_2 = \{x \mid f^2(x) = x\}$. If C_2 is a closed and connected part of $\mathbb{R}$ and if C_1 is a point with $C_2 \setminus C_1$ around it or if $C_2 \setminus C_1$ is empty, then continue with Step 2, else answer *no*.

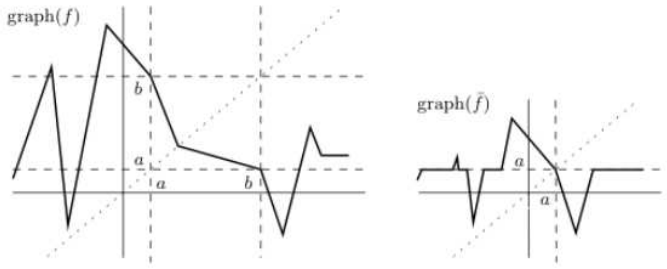


Figure 22: The contraction $\tilde{f}$ of a function f.

Step 2. If C_2 is $\mathbb{R}$, answer *yes*, else compute the function $\tilde{f}$ and decide whether $\tilde{f}$ is NILPOTENT and return the answer.

In this algorithm, $\tilde{f}$ is obtained from f by contracting the part of f that consists of of points of period 1 or 2, to one point. This is illustrated in Figure 22.

We remark that $f^k(\mathbb{R}) = C_1 \cup C_2$ if and only if $\tilde{f}^k(\mathbb{R}) = \{0\}$.

We now illustrate the working of the algorithm TERMINATE.

First, consider the a non-termininting function of Figure 20. Here, we have $C_1 = \{(0,0),(1,1)\}$ and $C_2 = \emptyset$ and the algorithm answers *no* since $C_1 \cup C_2$ is disconnected.

Next, consider the termininting function of Figure 21. Here we have $C_1 = \{(0,0)\}$ and $C_2 = \emptyset$, and thus $C_1 \cup C_2$ is closed and connected and since $\tilde{f} = f$is nilpotent the algorithm answers *yes*.

Small extensions of first-order logic with transitive closure The above decidability result concerning termination of functions, inspires the following extension of first-order logic with transitive closure restricted to be applied to the graphs of continuous functions $\mathbb{R}^k \to \mathbb{R}^k$. More precisely, let $\psi(\vec{x},\vec{y})$ be a formula in transitive-closure logic (with or without stop condition). Consider the sentence

$$\gamma_\psi = \gamma^1_\psi \wedge \gamma^2_\psi,$$

where γ^1_ψ expresses that $\psi(\vec{x},\vec{y})$ defines the graph of a function from $\mathbb{R}^k$ to $\mathbb{R}^k$ and γ^2_ψ expresses that $\psi(\vec{x},\vec{y})$ defines a continuous function graph.

Then it is easy to see that $\psi(\vec{x},\vec{y})$ terminates on input A if and only if γ_ψ terminates on A.

Now, we define *CF-transitive-closure logic*, an extension of first-order logic with a transitive-closure operator that is restricted to graphs of continuous functions $\mathbb{R}^k \to \mathbb{R}^k$.

More formally, CF-transitive-closure logic is the fragment of transitive-closure logic (respectively with stop condition) where expressions of the form

$$[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y}) \wedge \gamma_\psi](\vec{s},\vec{t})$$

(respectively $[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y}) \wedge \gamma_\psi \mid \sigma](\vec{s},\vec{t})$) are allowed, with γ_ψ a sentence that expresses that $\psi(\vec{x},\vec{y})$ defines the graph of a continuous function $\mathbb{R}^k \to \mathbb{R}^k$.

From earlier undecidability results we immediately get

Corollary 5. *It is undecidable whether a given formula in CF-transitive-closure logic with stop condition, where the transitive closure is restricted to work on binary relations, terminates on a given input database.*

Without stop conditions, we have decidability, however.

Theorem 15. *It is decidable whether a given formula in CF-transitive-closure logic (without stop conditions), restricted to binary relations, terminates on a given input database. Moreover, this decision procedure is expressible in this language.*

So, for every formula $[\mathrm{TC}_{x;y}\,\psi(x,y)](s,t)$ in CF-transitive-closure logic (without stop conditions), restricted to binary relations, there is a a formula τ_ψ in the same language that expresses that the formula terminates on a given input database (also τ_ψ depends on the input!). We call τ_ψ the *termination guard* of the formula $[\mathrm{TC}_{x;y}\,\psi(x,y)](s,t)$.

Now, we can define *GCF-transitive-closure logic*, the guarded fragment of CF-transitive-closure logic (without stop conditions), restricted to binary relations, in which only transitive-closure expressions of the form

$$[\mathrm{TC}_{x;y}\,\psi(x,y) \wedge \tau_\psi](s,t)$$

are allowed.

We end this section, with an expressivity result concerning these last two languages.

Theorem 16. *In GCF-transitive-closure logic, every query terminates on all possible input datasets and all terminating queries of CF-transitive-closure logic are expressible in GCF-transitive-closure logic.*

Furthermore, GCF-transitive-closure logic is more expressive than first-order logic on finite spatial datasets.

For what concerns the last part of the theorem, we remark that the query Q_{int} on 1-dimensional datasets S that expresses "Is S a singleton that contains a natural number?" is expressible in GCF-transitive-closure logic but not in first-order logic.

The results in this section can be found in [7].

11 Some concluding remarks on transitive-closure logics

One of the motivations to study these different transitive-closure logics, is to compare their expressive power and to establish which languages are computationally complete on linear or arbitrary (semi-algebraic) datasets.

It is not clear whether transitive-closure logic with stop conditions is more expressive than transitive-closure logic without stop conditions. In particular, it is not clear whether transitive-closure logic without stop conditions is also computationally complete on linear spatial datasets.

We also remark that for CF-transitive-closure logic without stop condition, termination is decidable and for CF-transitive-closure logic with stop condition termination is not decidable. This does not separate these languages, however (because equivalence is undecidable).

We also remark that many results on semi-algebraic functions also hold for arbitrary real closed fields. But termination of continuous semi-algebraic functions $f : R \to R$ for arbitrary real closed fields R is not first-order expressible. (for $\mathbb{R}$ the proof relies on Bolzano-Weierstrass).

References

[1] M. Benedikt, B. Kuijpers, C. Löding, J. Van den Bussche, T. Wilke. A characterization of first-order topological properties of planar spatial data. *Journal of the ACM*, 53(2):273–305, 2006.

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

[3] M. Coste. Ensembles semi-algébriques. In *Géometrie algébrique réelle et formes quadratiques*, Lectures Notes in Mathematics vol. 959, Springer, 1982.

[4] H.-D. Ebbinghaus, J. Flum. *Finite Model Theory*, second edition. Springer, 1999.

[5] Floris Geerts, Bart Kuijpers. Expressing topological connectivity of spatial databases. *Proceedings of 7th International Workshop on Database Programming Languages (DBPL'99)*, Lecture Notes in Computer Science 1949, 221-235, Springer-Verlag, 2000.

[6] Floris Geerts, Bart Kuijpers. Linear approximation of planar spatial databases using transitive-closure logic. *Proceedings of the 19th Symposium on Principles of Database Systems (PODS'00)* 126-135, ACM Press, 2000.

[7] Floris Geerts, Bart Kuijpers. On the Decidability of Termination of Query Evaluation in Transitive-Closure Logics for Polynomial Constraint Databases. *Theoretical Computer Science*, Vol. 336, Nr. 1, 125-151, 2005.

[8] Floris Geerts, Bart Kuijpers and Jan Van den Bussche. Linearization and completeness results for terminating transitive closure queries on spatial databases. *SIAM Journal on Computing*, Volume 35, Issue 6, 1386-1439, 2006.

[9] E. Grädel et al. *Finite Model Theory and Its Applications*. Springer, 2007.

[10] M. Grohe, L. Segoufin. On first-order topological queries. *ACM Transactions on Computational Logic*, 3(3):336–358, 2002.

[11] S. Grumbach, J. Su. Queries with arithmetical constraints. *Theoretical Computer Science*, 173(1):151–181, 1997.

[12] M. Gyssens, J. Van den Bussche, D. Van Gucht. Complete geometric query languages. *Journal of Computer and System Sciences*, 58(1):54–68, 1999.

[13] S. Kreutzer. Operational semantics for fixed-point logics on constraint databases. *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'01)*, Lecture Notes in Computer Science 2250, 470–484. Springer-Verlag, 2001.

[14] B. Kuijpers, J. Paredaens, J. Van den Bussche. On topological elementary equivalence of closes semi-algebraic sets in the real plane. *Journal of Symbolic Logic*, 65(4):1530–1555, 2000.

[15] Bart Kuijpers, Jan Paredaens, Marc Smits, Jan Van den Bussche. Termination properties of spatial Datalog programs *Proceedings of "Logic in Databases" (LID'96)*, Lecture Notes in Computer Science 1154, 101-116, 1996.

[16] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[17] R. McNaughton, S. Papert. *Counter-Free Automata*. MIT Press, 1971.

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

[19] L. Segoufin, V. Vianu. Querying spatial databases via topological invariants. *Journal of Computer and System Sciences*, 61(2):270–301, 2000.

[20] W. Schwabhäuser, W. Szmielew, A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, 1983.

[21] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume 3. Springer, 1997.