

## Real algebraic geometry and constraint databases

Non Peer-reviewed author version

GEERTS, Floris & KUIJPERS, Bart (2007) Real algebraic geometry and constraint databases. In: AIELLO, Marco & van BENTHEM, Johan & PRATT-HARTMANN, Ian (Ed.) Handbook of Spatial Logics, p. 799-856..

Handle: <http://hdl.handle.net/1942/7903>

## Chapter 1

# REAL ALGEBRAIC GEOMETRY AND CONSTRAINT DATABASES

Floris Geerts

*Hasselt University, Transnational University of Limburg & University of Edinburgh*

Bart Kuijpers

*Hasselt University & Transnational University of Limburg*

**Second Reader**

Peter Revesz

*University of Nebraska–Lincoln*

### 1. From the relational database model to the constraint database model

The constraint database model can be seen as a generalization of the classical relational database model that was introduced by Codd in the 1970s to deal with the management of alpha-numerical data, typically in business applications (?). A relational database can be viewed as a finite collection of tables or relations that each contain a finite number of tuples.

Figure 1.1 shows an instance of a relational database that contains the two relations **Beer** and **Pub**. This database contains tourist information about beers and the pubs where they are served. It also contains the location of the pubs, given in  $(x, y)$ -coordinates on some tourist map. Each relation contains a finite number of tuples. A relational database is usually modeled following a database *schema*. A schema contains information on the relation names and on the names of the attributes appearing in relation. In this example, the attributes of **Beer** are *Name*, *Pub*, *City* and *Postal code*. The complete schema of

<b>Beer</b>			
<i>Name</i>	<i>Pub</i>	<i>City</i>	<i>Postal code</i>
Duvel	De Muze	Antwerpen	2000
Hoegaarden	Villicus	Hasselt	3500
Geuze	La Bécasse	Brussel	1000
...	...	...	...

<b>Pub</b>		
<i>Pub</i>	<i>x</i>	<i>y</i>
De Muze	16	10
Villicus	16.1	14
La Bécasse	10.4	12.3
...	...	...

Figure 1.1. An example of a relational database consisting of the two relations **Beer** and **Pub**.

the relational database of Figure 1.1 could be written as **Beer**(*Name*, *Pub*, *City*, *Postal code*), **Pub**(*Pub*, *x*, *y*).

The *x* and *y* attributes of the relation **Pub** have a geometric or geographic interpretation. But values of these attributes can simply be stored as numbers, as is usually done in business databases. A tourist could consult this database to find out the locations of pubs where his/her preferred beers are served. First-order logic based languages (and their commercial versions, such as SQL) are used in the relational database model, to formulate queries like this. The vocabulary of these logics typically contains the relation names appearing in the schema of the input database. For instance, the first-order formula

$$\varphi(x, y) = \exists p \exists c \exists p' (\mathbf{Beer}(\text{Westvleteren}, p, c, p') \wedge \mathbf{Pub}(p, x, y))$$

when interpreted over the database of Figure 1.1, defines the (*x*, *y*)-coordinates of the location of the pubs where they serve my favorite beer.

But a tourist is usually also given more explicit geographic information, e.g., in the form of maps such as the one depicted in Figure 1.2 and he/she typically wants to ask questions that combine spatial and alphanumeric information, such as “*Where in Flanders, not too far from the river Scheldt, can I drink a Duvel?*”

In the relational database model, it is difficult to support queries like this one. Unlike the locations of pubs, the locations of rivers or regions would require the storage of infinitely many *x*- and *y*-coordinates of points. Storing infinitely many tuples is not possible and in computer

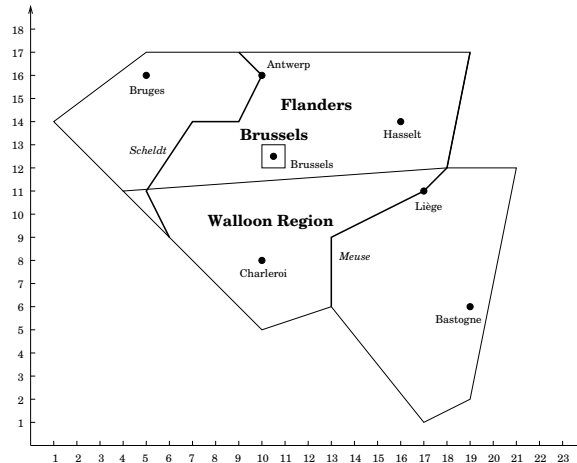


Figure 1.2. Spatial information map of Belgium.

science it is customary to find *finite* representations of even infinite sets or objects.

In the 1980s, extensions of the relational model have been proposed with special-purpose data types and operators. Data types like “polyline” and “polygon” were introduced to support, e.g., the storage of rivers and regions. Ad-hoc operations like intersection of polygons were added to popular query languages such as SQL. Later on, spatial database theory and technology has developed towards more sophisticated data models and more elegant query formalisms supported by, for example, appropriate indexing techniques. For an overview of the developments in spatial databases in the last two decades, we refer to (?).

Looking again at the polylines and polygons in Figure 1.2, we may remark that there are other finite ways to store them, besides the indirect method of storing their corner points. Indeed, each line segment can be described by linear equations (equalities and inequalities). Also polygonal figures can be described by combinations of linear inequalities. This description is more explicit than listing the corner points. If we agree that the combinations of linear equations may appear in the tuples of the relations of a database under a *geometric* attribute name, the spatial information displayed on the map of Belgium, which could be categorized into region, city, and river information, could be captured in a database with the three relations **Regions**, **Cities**, **Rivers**. Each of these relations has *Name* and *Geometry* as attributes, where the latter can be viewed as having an *x*- and a *y*-component. *Name* is a traditional alpha-numeric attribute and *Geometry* has a spatial or geometric inter-

**Cities**

<i>Name</i>	<i>Geometry(x, y)</i>
Antwerp	$(x = 10) \wedge (y = 16)$
Bastogne	$(x = 19) \wedge (y = 6)$
Bruges	$(x = 5) \wedge (y = 16)$
Brussels	$(x = 10.5) \wedge (y = 12.5)$
Charleroi	$(x = 10) \wedge (y = 8)$
Hasselt	$(x = 16) \wedge (y = 14)$
Liège	$(x = 17) \wedge (y = 11)$

**Rivers**

<i>Name</i>	<i>Geometry(x, y)</i>
Meuse	$((y \leq 17) \wedge (5x - y \leq 78) \wedge (y \geq 12)) \vee$ $((y \leq 12) \wedge (x - y = 6) \wedge (y \geq 11)) \vee$ $((y \leq 11) \wedge (x - 2y = -5) \wedge (y \geq 9)) \vee$ $((y \leq 9) \wedge (x = 13) \wedge (y \geq 6))$
Scheldt	$((y \leq 17) \wedge (x + y = 26) \wedge (y \geq 16)) \vee$ $((y \leq 16) \wedge (2x - y = 4) \wedge (y \geq 14)) \vee$ $((x \leq 9) \wedge (x \geq 7) \wedge (y = 14)) \vee$ $((y \leq 14) \wedge (-3x + 2y = 7) \wedge (y \geq 11)) \vee$ $((y \leq 11) \wedge (2x + y = 21) \wedge (y \geq 9))$

**Regions**

<i>Name</i>	<i>Geometry(x, y)</i>
Brussels	$(y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)$
Flanders	$(y \leq 17) \wedge (5x - y \leq 78) \wedge (x - 14y \leq -150) \wedge$ $(x + y \geq 45) \wedge (3x - 4y \geq -53) \wedge (\neg((y \leq 13) \wedge$ $(x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)))$
Walloon Region	$((x - 14y \geq -150) \wedge (y \leq 12) \wedge (19x + 7y \leq 375) \wedge$ $(x - 2y \leq 15) \wedge (5x + 4y \geq 89) \wedge (x \geq 13)) \vee$ $((-x + 3y \geq 5) \wedge (x + y \geq 45) \wedge$

Figure 1.3. Representation of the spatial database of Belgium shown in Figure 1.2.

pretation. Of course we could include more thematic information, e.g., we could add to the **City** relation the number of inhabitants.

The database instance with this schema, corresponding to the map shown in Figure 1.2, is given in Figure 1.3.

The geometric components of the relations in Figure 1.3 are described using linear equalities, linear inequalities and boolean combinations thereof, i.e., using  $\wedge$  (conjunction),  $\vee$  (disjunction) and  $\neg$  (negation). Figures that can be described in this way are sometimes referred to as *semi-linear set* figures.

One of the most important application areas of spatial databases is Geographic Information Systems (GIS), where in most cases polygonal-

shaped geometric figures are considered. In most cases this data resides in the two-dimensional plane or in the three-dimensional space (?). Indeed, in GIS, information is mostly linear of nature, but in other applications, like CAD-CAM, or medical imaging we can find spatial figures that are not linear. Using *polynomial* equalities and inequalities rather than just linear ones, gives us wider modeling capabilities. Figure 1.4 gives an example of a figure in the plane that can be described by the following combination of polynomial (in)equalities:

$$(x^2/25 + y^2/16 \leq 1) \wedge (x^2 + 4x + y^2 - 2y \geq -4) \\ \wedge (x^2 - 4x + y^2 - 2y \geq -4) \wedge ((x^2 + y^2 - 2y \neq 8) \vee (y > -1)).$$

This figure is described by a formula containing two variables, namely  $x$  and  $y$ , representing the coordinates of points in  $\mathbb{R}^2$  (Here,  $\mathbb{R}$  denotes the set of real numbers).

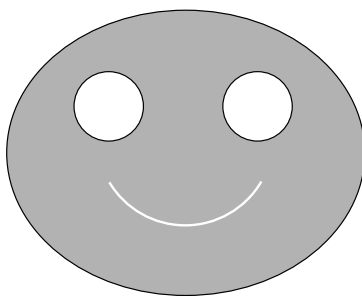


Figure 1.4. An example of a semi-algebraic set in  $\mathbb{R}^2$ .

Figures that can be modeled by polynomial inequalities are known, in mathematics, as *semi-algebraic sets* and their geometric and topological properties are well-studied in real algebraic geometry (?).

Semi-algebraic sets are, together with classical alpha-numeric data, the basic ingredients in *constraint databases*. As we have seen above in Figure 1.2, these sets appear in a constraint database by means of a defining formula. In this sense, the constraint database model is a generalization of the relational database model.

Similar to the classical relational database model, first-order logic can be used to formulate queries in the constraint model. Semi-algebraic sets are described by boolean combinations of (linear) polynomial inequalities, which basically are quantifier-free formulas in first-order logic over the reals. This logic has addition and multiplication as functions, order as relation and zero and one as constants. In the constraint database model, an extension of this logic with predicates to address the relations in the input database, is used as a basic logical query language.

This logic turns out to be a language in which a lot of relevant spatial database queries can be formulated. For example, the query “*Where in Flanders, not too far from the river Scheldt, can I drink a Duvel?*” can be expressed by the formula

$$\begin{aligned} \varphi(x, y) = & \mathbf{Regions}(\text{Flanders}, x, y) \wedge \\ & \exists x' \exists y' (\mathbf{Rivers}(\text{Scheldt}, x', y') \wedge (x - x')^2 + (y - y')^2 < 1) \wedge \\ & \exists p \exists c \exists p' (\mathbf{Pubs}(p, x, y) \wedge \mathbf{Beer}(\text{Duvel}, p, c, p')) \end{aligned}$$

Here, we translate “not too far from the river Scheldt” by “at most distance 1 from the some point of the Scheldt”. We remark that some variables in this expression are assumed to range over finite domains (namely  $p, c, p'$ ), but others range over the real numbers (namely  $x, y, x'$  and  $y'$ ). Nevertheless, it turns out that queries expressed by first-order formulas like this one, can be effectively evaluated on constraint databases. In our example the output is a two-dimensional geometric object and the query evaluation algorithm guarantees that it also can be described by a boolean combination of polynomial inequalities.

The ideas presented above are at the basis of the constraint database model. The basic idea is to extend or generalize the relational model and not only to allow finite relations, but also finitely representable relations.

We remark that the constraint database model was introduced by Kanellakis, Kuper and Revesz (?). It has received a lot of research attention since. An overview of research results in this field can be found in (?) and Revesz has written a textbook on the subject (?).

**Overview.** This chapter is organized as follows. In Section 1.2, we describe the constraint database model with its data models and basic query languages. Section 1.3 gives an overview of some definitions and results in real algebraic geometry, that will be used further on. In Section ??, we discuss query evaluation in the constraint database model through quantifier elimination. We also outline some quantifier elimination algorithms there. Section ?? is devoted to the expressive power of first-order logic over the reals as a query language for constraint databases. Topological queries get special attention. Finally, in Section ??, we discuss some more powerful query languages for constraint databases that are extensions of first-order logic, with transitive closure operators, with while-loop and with topological operators.

## 2. Constraint data models and query languages

In this section, we define the logics  $\text{FO}(+, \times, <, 0, 1)$ , i.e., *first-order logic with polynomial constraints*, and  $\text{FO}(+, <, 0, 1)$ , i.e., *first-order logic with linear constraints*, and show how they form the basis of the constraint approach in both the modeling and querying of spatial data. More specifically, we introduce the *polynomial* and *linear constraint model* and extend  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  to query languages for the respective models.

### 2.1 The logics $\text{FO}(+, \times, <, 0, 1)$ and $\text{FO}(+, <, 0, 1)$

Let  $(+, \times, <, 0, 1)$  be a so-called *vocabulary* with two functions symbols of arity two ( $+$  and  $\times$ ), one predicate symbol of arity two ( $<$ ), and two constant symbols (0 and 1). In the constraint model, this vocabulary will be interpreted on the *real field*, i.e., the structure consisting of the set of real numbers,  $\mathbb{R}$ , equipped with the standard addition, multiplication, and order.

We define  $\text{FO}(+, \times, <, 0, 1)$  as the *first-order logic over the vocabulary*  $(+, \times, <, 0, 1)$ . We build formulas in  $\text{FO}(+, \times, <, 0, 1)$  in the standard way: a *term*  $t$  in  $\text{FO}(+, \times, <, 0, 1)$  is either a *variable*  $x_i$ ; a constant (0 or 1); or of the form  $t + t'$  or  $t \times t'$  for terms  $t$  and  $t'$ . In other words, terms are polynomials with integer coefficients. Next, *atomic formulas* in  $\text{FO}(+, \times, <, 0, 1)$  are formulas of the form  $t = t'$  or  $t < t'$  for terms  $t$  and  $t'$ . Finally, formulas in  $\text{FO}(+, \times, <, 0, 1)$  are built from atomic formulas by using the *boolean connectives* ( $\wedge$ ,  $\vee$ , or  $\neg$ ) and quantifiers ( $\forall x_i$  or  $\exists x_i$ ). A variable is called *free* in a formula if it is not bounded by a quantifier. We denote by  $\varphi(x_1, \dots, x_n)$  that the  $\text{FO}(+, \times, <, 0, 1)$  formula  $\varphi$  has  $n$  free variables  $x_1, \dots, x_n$ . A formula without any free variables is called a *sentence*. A formula without quantifiers is called *quantifier-free*.

Similarly, we define  $\text{FO}(+, <, 0, 1)$  as the restriction of  $\text{FO}(+, \times, <, 0, 1)$  in which formulas are constructed from terms which do not use multiplication (i.e., formulas without  $\times$ ). In other words, the terms in  $\text{FO}(+, <, 0, 1)$  are polynomials with integer coefficients of degree at most one. We also say that  $\text{FO}(+, <, 0, 1)$  is the *first-order logic over the vocabulary*  $(+, <, 0, 1)$ .

We define the *satisfaction* of a formula  $\varphi(x_1, \dots, x_n)$  in  $\text{FO}(+, \times, <, 0, 1)$  by real numbers  $r_1, \dots, r_n \in \mathbb{R}$ , denoted by

$$(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_n),$$

inductively on the structure of  $\varphi$ :



- $(\mathbb{R}, +, \times, <, 0, 1) \models (t = t')(r_1, \dots, r_n)$  if  $t(r_1, \dots, r_n) = t'(r_1, \dots, r_n)$ ;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (t < t')(r_1, \dots, r_n)$  if  $t(r_1, \dots, r_n) < t'(r_1, \dots, r_n)$ ;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (\neg\varphi)(r_1, \dots, r_n)$  if  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_n)$  does not hold;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (\varphi \wedge \psi)(r_1, \dots, r_n)$  if  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_n)$  and  $(\mathbb{R}, +, \times, <, 0, 1) \models \psi(r_1, \dots, r_n)$ ;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (\varphi \vee \psi)(r_1, \dots, r_n)$  if  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_n)$  or  $(\mathbb{R}, +, \times, <, 0, 1) \models \psi(r_1, \dots, r_n)$ ;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (\forall x_n \varphi)(r_1, \dots, r_{n-1})$  if for all elements  $r \in \mathbb{R}$ ,  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_{n-1}, r)$ ; and
- $(\mathbb{R}, +, \times, <, 0, 1) \models (\exists x_n \varphi)(r_1, \dots, r_{n-1})$  if there exists an element  $r \in \mathbb{R}$ ,  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, \dots, r_{n-1}, r)$ .

As described above, in the constraint model, the satisfaction of formulas in  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  is defined with respect to the real field  $\mathbb{R}$ . However, any mathematical structure which interprets the vocabularies  $(+, \times, <, 0, 1)$  or  $(+, <, 0, 1)$  can be used instead.

**EXERCISE 1.1** Let  $\mathbb{N}$  denote the set of natural numbers equipped with the standard addition, multiplication and ordering between natural numbers.

- 1 Give a sentence  $\varphi_{\text{dense}}$  in  $\text{FO}(+, <, 0, 1)$  that expresses that the order  $<$  of  $\mathbb{R}$  is *dense*, i.e., for any two real numbers  $r_1$  and  $r_2$  such that  $r_1 < r_2$ , there exists a real number  $r_3$  such that  $r_1 < r_3 < r_2$ .
- 2 Verify  $\mathbb{R} \models \varphi_{\text{dense}}$  but  $\mathbb{N} \not\models \varphi_{\text{dense}}$ .

Of particular importance in the constraint model are the quantifier-free formulas in  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$ . As we will see in the next section, the representation of spatial objects by means of quantifier-free formulas is the basis of the data model in constraint databases. In Section ??, we show that both  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  admit *quantifier elimination*. In short, this means that any formula in  $\text{FO}(+, \times, <, 0, 1)$  (respectively  $\text{FO}(+, <, 0, 1)$ ) is equivalent to a quantifier-free formula in  $\text{FO}(+, \times, <, 0, 1)$  over  $\mathbb{R}$  (respectively in  $\text{FO}(+, <, 0, 1)$ ). Hence, we do not lose any generality by considering quantifier-free formulas only. As mentioned in the introduction, a (quantifier-free) formula represents a possible infinite set of points.

More specifically, they describe sets of points which correspond to *semi-algebraic sets*, in case of  $\text{FO}(+, \times, <, 0, 1)$ , and *semi-linear sets*, in case of  $\text{FO}(+, <, 0, 1)$  (see also Section 1.3).

**EXAMPLE 1.2** In Figure 1.4 of Section 1.1, the smiling face shows all pairs  $(r_1, r_2) \in \mathbb{R}^2$  that satisfy  $\varphi(x, y)$ , i.e.,  $(\mathbb{R}, +, \times, <, 0, 1) \models \varphi(r_1, r_2)$ , where  $\varphi(x, y)$  is the quantifier-free formula

$$x^2/25 + y^2/16 \leq 1 \wedge x^2 + 4x + y^2 - 2y \geq -4 \wedge \\ x^2 - 4x + y^2 - 2y \geq -4 \wedge (x^2 + y^2 - 2y \neq 8 \vee y > -1).$$

We remark that  $\varphi$  has two free variables and that it uses polynomials of degree at most two.

Apart from the modeling of spatial data, the logics  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  serve also as the basis of the standard *query languages* in the constraint model. We come back to this point in the next sections.

## 2.2 The polynomial constraint data model

First, we discuss the general polynomial constraint model which is based on  $\text{FO}(+, \times, <, 0, 1)$ . In the next section, we elaborate on the linear constraint model which uses  $\text{FO}(+, <, 0, 1)$ .

**The polynomial constraint data model.** A *database schema*  $\mathcal{S}$  is a finite set  $\{S_1, \dots, S_k\}$  of relation names. Each relation name  $S_i$  ( $i = 1, \dots, k$ ) is of some arity  $n_i$ , which is an integer. A *polynomial constraint relation instance* of  $S_i$ , or *constraint relation* of  $S_i$  for short, maps  $S_i$  to a quantifier-free formula  $\varphi_{S_i}(x_1, \dots, x_{n_i})$  with  $n_i$  free variables in the logic  $\text{FO}(+, \times, <, 0, 1)$ . A *(polynomial) constraint database instance over  $\mathcal{S}$*  consists of a set of constraint relations of  $S_1, \dots, S_k$ .

The *semantics* of a relation instance of  $S_i$ , denoted by  $I(S_i)$ , is the possible infinite (semi-algebraic) subset

$$\{(r_1, \dots, r_{n_i}) \in \mathbb{R}^{n_i} \mid (\mathbb{R}, +, \times, <, 0, 1) \models \varphi_{S_i}(r_1, \dots, r_{n_i})\}.$$

The semantics of a (polynomial) constraint database instance  $D$ , denoted by  $I(D)$ , over the schema  $\mathcal{S}$ , is the collection of semi-algebraic sets  $I(S_i)$ , with  $S_i$  a relation name appearing in  $\mathcal{S}$ .

**EXAMPLE 1.3** Let  $\mathcal{S} = \{S\}$ , where  $S$  is a binary relation name. A constraint database instance  $D$  over  $\mathcal{S}$  maps, for instance,  $S$  to the quantifier-free  $\varphi(x, y)$  given in Example 1.2. The semantics of  $D$  is the smiling face shown in Figure 1.4.

## EXERCISE 1.4

- 1 Give a formula describing (1) the smiling face of Figure 1.4 with the left eye closed (i.e., filled) and (2) a sad version of this face.
- 2 Give an instance of a polynomial constraint database containing two relations of arity 3 that contain respectively a representation of the Earth and Saturn (including its rings).

It is clear that the same semi-algebraic set can be represented by different formulas. Indeed, consider again Figure 1.4. Suppose that the description of the smiling face given in Example 1.2 is extended with the disjunct  $(x = 0 \wedge -1/2 \leq y \leq 1/2)$  (i.e., a vertical line segment), representing a nose. This new representation will not lead to the addition of new points in the smiling face, since all the points in the nose are already part of the face.

Two constraint relations of  $S$  and  $S'$  (i.e., formulas) are said to be *equivalent* if  $I(S)$  and  $I(S')$  are the same semi-algebraic set (i.e., if  $I(S) = I(S')$ ). Similarly, we say that two database instances are equivalent if their relations are pairwise equivalent.

REMARK 1.5 In the remainder of this chapter, we will interchangeably use constraint relation and semi-algebraic set, since these notions refer to the same objects, albeit from different perspectives.

**Database queries in the constraint model.** Before we explain how to use  $\text{FO}(+, \times, <, 0, 1)$  as a query language for the polynomial constraint model, we define what we mean by a query on a constraint database. In standard relational databases, a query is a (partial) function associating with each input database instance an output relation instance. In the constraint setting, however, there are two ways of looking at a query.

- First, similar to the relational setting, we can define a *k-ary query over a database schema  $\mathcal{S}$*  as a partial function which associates with a database instance  $I(D)$  (i.e., a collection of semi-algebraic sets), a semi-algebraic set in  $\mathbb{R}^k$ , where  $D$  is any database instance of  $\mathcal{S}$ .
- Second, we can also view a *k-ary query over  $\mathcal{S}$*  as a partial function associating with each database instance  $D$  (i.e., a collection of quantifier-free formulas), a quantifier-free formula in  $\text{FO}(+, \times, <, 0, 1)$  with  $k$  free variables.

We call the first type of query an *unrestricted query*; the second is called a *constraint query*. A constraint query clearly only makes sense if it maps two equivalent database instances to equivalent relation instances (i.e., equivalent quantifier-free  $\text{FO}(+, \times, <, 0, 1)$ -formulas). If a constraint query satisfies this property, we call a constraint query *consistent*. We remark that a consistent constraint query corresponds to a *unique* unrestricted query.

**EXAMPLE 1.6** Let  $\mathcal{S}$  consist of binary relation  $S$ . Consider the constraint query  $Q$  which maps any constraint relation of  $S$ , given by  $\varphi_S$ , to the highest degree of polynomials appearing in  $\varphi_S$ . This query is clearly not consistent. Indeed, let  $\varphi_S \equiv x^2 + y^2 = 1$  and  $\varphi'_S \equiv (x^2 + y^2)^2 = 1$ . Both formulas correspond to the same semi-algebraic set, i.e., the standard circle of radius 1. In contrast,  $Q$  returns 2 on input  $\varphi_S$ , whereas it returns 4 on input  $\varphi'_S$ .

In the following, when we refer to a *constraint database query*, we mean a consistent constraint query.

**The logic  $\text{FO}(+, \times, <, 0, 1)$  as a query language for polynomial constraint databases.**

In this section, we take a closer look at the standard query language for polynomial constraint databases which is an extension of  $\text{FO}(+, <, 0, 1)$  with predicates to address constraint relations that appear in the input database.

If we consider queries over a database input schema  $\mathcal{S} = \{S_1, \dots, S_k\}$ , then we can associate a query with a formula in the first-order logic over the vocabulary  $(+, \times, <, 0, 1, S_1, \dots, S_k)$ . Let  $\varphi(x_1, \dots, x_m)$  be such a formula over  $(+, \times, <, 0, 1, S_1, \dots, S_k)$ . Given a constraint database  $D$  over  $\mathcal{S}$ , we interpret  $\varphi(x_1, \dots, x_m)$  over the  $(\mathbb{R}, +, \times, <, 0, 1)$ , extended with the semi-algebraic sets,  $I(S_1), \dots, I(S_k)$  as given by  $D$ . More specifically, the  $m$ -ary answer set of  $\varphi(x_1, \dots, x_m)$  is defined as

$$\{(r_1, \dots, r_m) \in \mathbb{R}^m \mid (\mathbb{R}, +, \times, <, 0, 1, I(S_1), \dots, I(S_k)) \models \varphi(r_1, \dots, r_m)\}.$$

We also write the above for short as

$$\{(r_1, \dots, r_m) \in \mathbb{R}^m \mid (\mathbb{R}, D) \models \varphi(r_1, \dots, r_m)\}.$$

It is clear that equivalent databases result in the same answer set. We say that  $\varphi$  *expresses* the corresponding (unique) unrestricted query. In the sequel, we refer to these extensions of  $\text{FO}(+, \times, <, 0, 1)$  simply by  $\text{FO}(+, \times, <, 0, 1)$  if the input schema is clear from the context or irrelevant.

An important property of any query language is that it is *closed*, i.e., the result of query should admit a representation in the same data

model as the source relations. In particular, for  $\text{FO}(+, \times, <, 0, 1)$  to be closed it should be the case that the result is a quantifier-free formula in  $\text{FO}(+, \times, <, 0, 1)$  again. However, since  $\text{FO}(+, \times, <, 0, 1)$  admits quantifier-elimination and by the way  $\text{FO}(+, \times, <, 0, 1)$ -formulas are evaluated, this requirement is satisfied (see also Section ??).

In Section 1.1, we gave examples of queries expressed in  $\text{FO}(+, \times, <, 0, 1)$ . We give some more examples here.

EXAMPLE 1.7 Let  $Q_{\text{bounded}}$  be the unrestricted query which returns true if and only if the input semi-algebraic set in  $\mathbb{R}^2$  is bounded. In the first-order logic over  $(+, \times, <, 0, 1, S)$ , where  $S$  is a binary relation name, the sentence

$$\exists \varepsilon (\varepsilon \neq 0 \wedge \forall x \forall y (S(x, y) \rightarrow x^2 + y^2 < \varepsilon^2))$$

expresses  $Q_{\text{bounded}}$ .

EXAMPLE 1.8 Let  $Q_{\text{interior}}$  be the query that returns all points of any input semi-algebraic set in  $\mathbb{R}^2$  that have a neighborhood that is completely in the semi-algebraic set. Hence,  $Q_{\text{interior}}$  returns the *topological interior* of a semi-algebraic set in  $\mathbb{R}^2$ . This query can be expressed as

$$\exists r \forall x' \forall y' (r \neq 0) \wedge ((x - x')^2 + (y - y')^2 < r^2 \rightarrow S(x', y')).$$

We remark that this formula has two free variables, so it defines a semi-algebraic set in  $\mathbb{R}^2$ .

#### EXERCISE 1.9

- 1 Express in  $(+, \times, <, 0, 1, S)$  the queries returning the topological closure and topological boundary of any input semi-algebraic set in  $\mathbb{R}^2$ .
- 2 Give an  $(+, \times, <, 0, 1, S)$ -sentence that expresses the query  $Q_{\text{finite}}$  which returns true if and only if the input semi-algebraic set in  $\mathbb{R}^2$  consists of finite number of points.
- 3 Give an  $(+, \times, <, 0, 1, S)$ -sentence that expresses the query  $Q_{\text{sphere}}$  which returns true if and only if the input semi-algebraic set in  $\mathbb{R}^3$  is a sphere.
- 4 Give a sentence over  $(+, \times, <, 0, 1, S_1, S_2)$  that expresses the query  $Q_{\text{intersect}}$  which returns true if and only if the two input semi-algebraic sets intersect.

In the next section, we discuss the expressive power of the query language  $\text{FO}(+, \times, <, 0, 1)$ . For the moment, let us merely say that it is

“rather limited.” Topological queries such as the topological interior are expressible in this logic, but we will see in Section ?? that important queries are not expressible in  $\text{FO}(+, \times, <, 0, 1)$ . More specifically, we will see that the query that expresses that a spatial database is *topologically connected* is not expressible. Due to the importance of this query in the spatial database practice, many efforts to extend  $\text{FO}(+, \times, <, 0, 1)$  to richer query languages exist, some of which we discuss in Section ??.

### 2.3 The linear constraint model: an application in Geographic Information Systems

Next, we discuss the linear constraint model, which is less expressive than the polynomial constraint model (as we will illustrate in Section ??), but nevertheless powerful enough to model applications like Geographic Information Systems, or GIS for short.

**The linear constraint data model.** Since we emphasize the GIS aspect of the linear model here, we will also combine linear spatial information with classical alpha-numeric information, as is customary in the GIS practice. Therefore, for the sake of illustrating the suitability for GIS, we consider more general database schemas and instances in this section.

Also, the linear constraint database model can be seen as based on the relational model. Moreover, linear constraint databases also require a lot of traditional database capabilities. In particular, if the linear constraint database consists purely of non-spatial flat relations, it degenerates into a traditional database for which the relational model offers a well-accepted representation.

More formally, a *linear constraint database scheme*  $\mathcal{S}$  consists of a finite set of relation names  $S_1, \dots, S_k$ . Each relation name  $S_i$  ( $i = 1, \dots, k$ ) is of some type  $[n_i, m_i]$ , with  $n_i$  and  $m_i$  integers. A *linear constraint database instance* is a mapping that assigns a linear relation instance to each relation name appearing in the database scheme. A *linear relation instance* of  $S_i$ , also called a *linear relation* for short, is a finite set of linear tuples of type  $[n_i, m_i]$ . A *linear tuple* of type  $[n_i, m_i]$  is straightforwardly defined as a tuple of the form

$$(c_1, \dots, c_{n_i}, \varphi(x_1, \dots, x_{m_i}))$$

where  $c_1, \dots, c_{n_i}$  are thematic values, typically from some alpha-numeric domain  $U$  (for instance.,  $U$  could be the set of all strings over our alphabet and natural numbers) and  $\varphi(x_1, \dots, x_{m_i})$  is a quantifier-free formula in the logic  $\text{FO}(+, <, 0, 1)$  with  $m_i$  free variables.

The semantics of a linear tuple  $t = (c_1, \dots, c_{n_i}, \varphi(x_1, \dots, x_{m_i}))$  of type  $[n_i, m_i]$  is the possibly infinite subset of  $U^{n_i} \times \mathbb{R}^{m_i}$  defined as the Cartesian product  $\{(c_1, \dots, c_{n_i})\} \times A_i$ , in which  $A_i \subseteq \mathbb{R}^{m_i}$  is the semi-linear set

$$\{(r_1, \dots, r_{m_i}) \in \mathbb{R}^{m_i} \mid (\mathbb{R}, +, <, 0, 1) \models \varphi(r_1, \dots, r_{m_i})\}.$$

This subset of  $U^{n_i} \times \mathbb{R}^{m_i}$  can be interpreted as a possibly infinite  $(n_i + m_i)$ -ary relation, denoted  $I(t)$ . The semantics of a linear relation,  $S_i$ , denoted  $I(S_i)$ , is defined as  $I(S_i) = \bigcup_{t \in S_i} I(t)$ . Finally, the semantics of a linear spatial database,  $D$  over the schema  $\mathcal{S}$ , is the set of relations  $I(S_i)$  with  $S_i$  a linear relation name appearing in the schema  $\mathcal{S} = \{S_1, \dots, S_k\}$  of  $D$ .

For GIS, where spatial information is often modeled in either the *vector model* or the *raster model*, and combined with traditional alphanumeric information often stored in a relational database, the linear constraint model is powerful enough. Indeed, in the vector model usually three types of planar spatial objects are used, namely *points*, *polylines* and *polygons*. In the raster model, the plane  $\mathbb{R}^2$  is divided by a regular grid. Clearly, if we assume the grid to be finite, both types of data can be modeled in the linear constraint model.

**EXAMPLE 1.10** In Section 1.1, we introduced the example of a map containing information about Belgium, as illustrated in Figure 1.2. The spatial information displayed on the map of Belgium can be categorized into city, river, and region information. Therefore, we introduced three relations, each containing one of these spatial information sources (Figure 1.3). The relations **Cities**, **Rivers** and **Regions** are of type  $[1, 2]$  and model respectively points, polylines and polygons. Their thematic component contains names (or string information), whereas their spatial component contains formulas describing spatial features of Belgium. This example illustrated that the linear constraint model is suitable for GIS.

### **The logic $\text{FO}(+, <, 0, 1)$ as a query language for Geographic Information Systems.**

In this section, we take a closer look at the standard query language for linear constraint databases which is an extension of  $\text{FO}(+, <, 0, 1)$  with predicates to address linear constraint relations that appear in the input database. Because of the mixed presence of thematic and spatial information, this query language will be an extension of  $\text{FO}(+, <, 0, 1)$  in the sense of a two-sorted logic. More specifically, if we consider queries over a database input schema  $\mathcal{S} = \{S_1, \dots, S_k\}$ , we have, apart from the terms, formulas and quantifications possible in  $\text{FO}(+, <, 0, 1)$ , the following ingredients:

- apart from (real) variables  $x_1, x_2, \dots$  ranging over  $\mathbb{R}$ , we also have infinitely *thematic variables*  $v_1, v_2, \dots$  ranging over  $U$  and distinct from the set of real variables;
- we have atomic formulas of the form  $v_1 = v_2$ , with  $v_1$  and  $v_2$  thematic variables;
- we have atomic formulas of the form  $S_i(v_{i_1}, \dots, v_{i_{n_i}}; t_{j_1}, \dots, t_{j_{m_i}})$ , with  $S_i$  a relation name of type  $[n_i, m_i]$ ,  $v_{i_1}, \dots, v_{i_{n_i}}$  are thematic variables, and  $t_{j_1}, \dots, t_{j_{m_i}}$  are terms in  $\text{FO}(+, <, 0, 1)$ ; and
- universal and existential quantification of thematic variables.

In the following, we will refer to this extension of  $\text{FO}(+, <, 0, 1)$ , simply as  $\text{FO}(+, <, 0, 1)$ . Similar to the case of  $\text{FO}(+, \times, <, 0, 1)$ , a formula  $\varphi(v_1, \dots, v_n, x_1, \dots, x_m)$  in  $\text{FO}(+, <, 0, 1)$  expresses a constraint query of type  $[n, m]$ .

Finally, we shall give some typical example queries, illustrating the expressive power of  $\text{FO}(+, <, 0, 1)$ .

**EXAMPLE 1.11** An example of a (very simple) linear spatial query on the database in Example 1.3 is “*Find all cities that lie on a river and give their names and the names of the rivers they lie on.*” This query can be expressed by the following first-order formula:

$$\varphi(c, r) = \exists x \exists y (\mathbf{Cities}(c, x, y) \wedge \mathbf{Rivers}(r, x, y)).$$

This formula defines an output relation of type  $[2, 0]$ .

In all the remaining queries, we shall assume the input database consists of one relation  $S$  of type  $[0, 2]$ .

**EXAMPLE 1.12** The following  $\text{FO}(+, <, 0, 1)$ -sentence expresses  $Q_{\text{bounded}}$  (see Example 1.6):

$$\exists d \forall x \forall y (S(x, y) \rightarrow -d < x \wedge x < d \wedge -d < y \wedge y < d).$$

**EXAMPLE 1.13** Several topological properties of a semi-linear set can be expressed in  $\text{FO}(+, <, 0, 1)$ . For instance, the query  $Q_{\text{interior}}$  (see Example 1.7) is expressed by the  $\text{FO}(+, <, 0, 1)$  formula

$$\varphi(x, y) = \exists \varepsilon \forall x' \forall y' (\varepsilon \neq 0) \wedge (|x - x'| < \varepsilon \wedge |y - y'| < \varepsilon) \rightarrow S(x', y').$$

The formula  $\varphi(x, y)$  represents a semi-linear set in  $\mathbb{R}^2$ .

**EXERCISE 1.14**



- Express in  $\text{FO}(+, <, 0, 1)$  the queries returning the topological closure and topological boundary of input semi-linear sets in  $\mathbb{R}^2$ .
- Express that an input relation of type  $[0, 2]$  is finite. It helps to know that a semi-linear set is finite if and only if it consists of isolated points.
- Give a sentence that expresses that an input relation of type  $[0, 2]$  has a convex spatial component. This is a rather tricky question. It helps to know that a semi-linear set  $A$  is convex if for any two points  $p$  and  $q$  belonging to  $A$  also  $\frac{p+q}{2}$  belongs to  $A$ .

In spite of all this,  $\text{FO}(+, <, 0, 1)$  cannot be considered as a fully adequate query language for practical purposes. More specifically, there are very simple queries which are not expressible in  $\text{FO}(+, <, 0, 1)$ , which are expressible in  $\text{FO}(+, \times, <, 0, 1)$ . We return to this issue in Section ??.

### 3. Introduction to real algebraic geometry

In this section, we define and discuss semi-algebraic and semi-linear sets and review some well-known properties of these sets. We are interested in sets which are situated in the  $n$ -dimensional Euclidean space  $\mathbb{R}^n$ .

An excellent introduction to real-algebraic geometry can be found in (?). Proofs of all the theorems given in this section can be found there. More advanced is the standard book in the field (?) and for a more algorithmic point of view we refer to (?). An interesting book covering many other aspects of real algebraic geometry is (?). On a very advanced level, investigations of real-algebraic geometry in terms of constructible sets, real spectra, and spaces of orderings can be found in (?).

Finally, the generalization of real-algebraic geometry to so-called o-minimal geometry is described in (?). An excellent book on o-minimal structures is (?). Interestingly, many results from constraint databases described in this chapter can be generalized to the o-minimal setting. We refer to the standard book on constraint databases for more details (?).

#### 3.1 Semi-algebraic sets and their basic properties

**Definition of semi-algebraic sets.** A *semi-algebraic subset* of  $\mathbb{R}^n$  is a subset of points  $\vec{x} = (x_1, \dots, x_n)$  in  $\mathbb{R}^n$  satisfying a Boolean combination (expressed by disjunction, conjunction and negation—or in set-theoretic terms by union, intersection, and complement) of polyno-

mial equations and inequalities with integer coefficients. It is easy to see that every semi-algebraic set in  $\mathbb{R}^n$  is the finite union of sets of the form

$$\{\vec{x} \in \mathbb{R}^n \mid f(\vec{x}) = 0, \quad g_1(\vec{x}) > 0, g_2(\vec{x}), \dots, g_\ell(\vec{x}) > 0\},$$

where  $f, g_1, \dots, g_\ell$  are multivariate polynomials in the variables  $x_1, \dots, x_n$  with integer coefficients. A *semi-linear subset of  $\mathbb{R}^n$*  is a semi-algebraic subset which is described by multivariate polynomials of degree at most one (i.e., linear multivariate polynomials).

REMARK 1.15 It is easy to see that the class of semi-algebraic sets defined above coincides with the class of sets represented by quantifier-free formulas in  $\text{FO}(+, \times, <, 0, 1)$ . We therefore are free to choose either of the two representations. We more often use the representation in terms of quantifier-free formulas.

EXAMPLE 1.16 In the introductory section we have already given an example of semi-algebraic sets (see, e.g., Figure 1.4 ). Semi-algebraic sets can be used to model various spatial situations, but also spatio-temporal phenomena, as is illustrated in Figure 1.5. Here a potential scene from Star Trek is depicted in which the starship Enterprise fires a photon torpedo. This scene plays in the three-dimensional  $(x, y, t)$  space, where  $x$  and  $y$  are spatial coordinates and  $t$  represents a time coordinate. The star ship remains at a constant position in space and can therefore be described by some fixed formula

$$\varphi_{\text{Enterprise}}(x, y, t) = ((x^2 + y^2 = 1) \vee (x^2 + y^2 = (1/4)^2) \vee \dots)$$

in which  $t$  does not appear. A fired photon torpedo follows the dotted line (between the moments  $t = 0$  and  $t = 1$ ) and then explodes (depicted as increasing dotted circles, between  $t = 1$  and  $t = 2$ ). At the bottom of Figure 1.5 three frames of the movie are shown: at  $t = 1/2, 1$  and  $2$ . The complete movie can be described by the set

$$\{(x, y, t) \in \mathbb{R}^2 \times \mathbb{R} \mid (\varphi_{\text{Enterprise}}(x, y) \wedge (0 \leq t \leq 2)) \vee ((y = 0 \wedge x = 4t) \wedge (0 \leq t \leq 1)) \vee (((x - 4)^2 + y^2 \leq (t - 1)) \wedge (1 < t \leq 2))\}.$$

**Basic properties of semi-algebraic sets.** The class of semi-algebraic sets is closed under finite unions, intersections and complements. Moreover, if  $A \subseteq \mathbb{R}^m$  and  $B \subseteq \mathbb{R}^n$  are semi-algebraic, then the cartesian product  $A \times B$  is a semi-algebraic subset of  $\mathbb{R}^{m+n}$ . A much

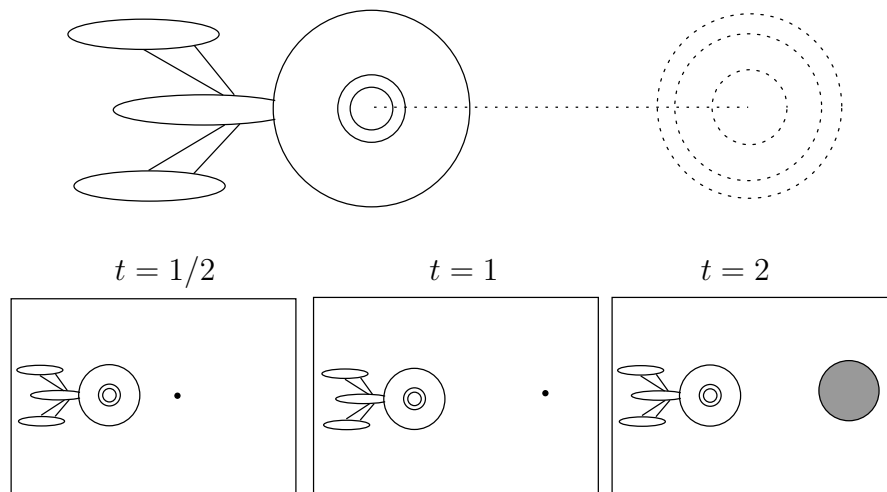


Figure 1.5. USS Enterprise firing a photon torpedo at a (cloaked) Klingon vessel.

deeper result is that the class of semi-algebraic sets is closed under projection as well:

**THEOREM 1.17 (TARSKI-SEIDENBERG)** *Let  $A$  be a semi-algebraic subset of  $\mathbb{R}^{n+1}$  and let  $\pi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$  be the projection on the first  $n$  coordinates. Then  $\pi(A)$  is a semi-algebraic set of  $\mathbb{R}^n$ .*

One may wonder whether all sets of  $\mathbb{R}^n$  are semi-algebraic. Already for  $n = 1$ , it can be shown that there are subsets that are not semi-algebraic. In fact, every semi-algebraic subset of  $\mathbb{R}$  is known to be a finite union of open intervals (possibly unbounded) and points. Figure 1.6 gives an example of a one-dimensional semi-algebraic set. It is the union of four open intervals (the leftmost being unbounded) and five points (three of which are isolated).



Figure 1.6. An example of a semi-algebraic set in  $\mathbb{R}$ .

From this property it follows that the set of natural numbers  $\mathbb{N}$  is not a semi-algebraic subset of  $\mathbb{R}$ .

**EXERCISE 1.18**

- 1 The reader may want to verify the above statement that the semi-algebraic sets of  $\mathbb{R}^1$  are the unions of finitely many points and open intervals.

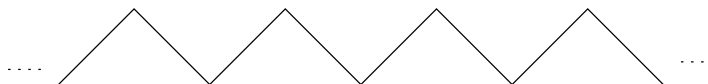


Figure 1.7. An example of a subset of  $\mathbb{R}^2$  that is not semi-algebraic.

- 2 Use the above result to show that the infinite zig-zag line shown in Figure ?? in  $\mathbb{R}^2$  is not semi-algebraic.
- 3 The topological closure  $\overline{A}$  of a set  $A$  of  $\mathbb{R}^n$  is the set of points  $\vec{p} \in \mathbb{R}^n$  for which any neighborhood  $U \subseteq \mathbb{R}^n$  of  $\vec{p}$  intersects  $A$ . Show that the topological closure  $\overline{A}$  of a semi-algebraic set  $A$  is also semi-algebraic.

Let  $A$  be a semi-algebraic set of  $\mathbb{R}^n$  and let  $f : A \rightarrow \mathbb{R}$  be a real-valued function. Then  $f$  is called a *semi-algebraic function* if its graph

$$\Gamma(f) = \{(\vec{x}, r) \in A \times \mathbb{R} \mid \vec{x} \in A \text{ and } r = f(\vec{x})\}$$

is a semi-algebraic set of  $\mathbb{R}^{n+1}$ .

**Curve selection.** The following result says that any point on the border of a semi-algebraic set can be connected to the set via a continuous curve.

**THEOREM 1.19 (CURVE SELECTION)** *Let  $A$  be a semi-algebraic set of  $\mathbb{R}^n$ , and let  $\vec{x} \in \overline{A} \setminus A$ . Then there exists a continuous semi-algebraic function  $\gamma : [0, 1] \rightarrow \mathbb{R}^n$  such that  $\gamma(0) = \vec{x}$  and  $\gamma((0, 1]) \subseteq A$ .*

A proof of this theorem can be found, e.g., in (?, Proposition 2.5.3).

**EXERCISE 1.20** A set  $A$  of  $\mathbb{R}^n$  is *connected* if there exists no open sets  $U, V$  of  $\mathbb{R}^n$  such that  $A = U \cup V$ ,  $U \cap \overline{V} = \emptyset$  and  $\overline{U} \cap V = \emptyset$ . Show that for a semi-algebraic  $A$  of  $\mathbb{R}^n$  being connected coincides with being semi-algebraically arc-connected (i.e., between any two points  $\vec{s}, \vec{t} \in A$  there exists a semi-algebraic function  $\gamma : [0, 1] \rightarrow \mathbb{R}^n$  such that  $\gamma(0) = \vec{s}$ ,  $\gamma(1) = \vec{t}$  and  $\gamma([0, 1]) \subseteq A$ ). (*Hint:* Make use of the curve selection theorem).

We remark that the curve selection theorem also holds when semi-algebraic is replaced by semi-linear.

### 3.2 Decompositions of semi-algebraic sets

**Topological decomposition.** The semi-algebraic sets of  $\mathbb{R}^1$  are characterized above as being finite unions of open intervals and points.

A similar characterization exists for semi-algebraic sets of  $\mathbb{R}^n$ , which we state here. A proof of this result can be found in (? , Theorem 2.3.6). We first recall the definition of an homeomorphism: an *homeomorphism*  $h$  between two sets  $X$  and  $Y$  is a continuous bijection which has continuous inverse. Two sets are called *homeomorphic* if there exists an homeomorphism between them.

**THEOREM 1.21** *Let  $A$  be a semi-algebraic subset of  $\mathbb{R}^n$ . Then  $A$  can be written as a finite union*

$$A = \bigcup_{i=0}^n \bigcup_{j=1}^{m_i} A_{ij},$$

where each  $A_{ij}$  is homeomorphic to the open cube  $(0,1)^i$ .

We remark that the *dimension* of  $(0,1)^i$  is  $i$ . So, this theorem states that any semi-algebraic set of  $\mathbb{R}^n$  can be decomposed into finite unions of objects that are from a topological point of view, open cubes of dimension lower or equal to  $n$ .

**EXERCISE 1.22**

- 1 Verify that this result corresponds, for  $n = 1$ , to the result mentioned earlier.
- 2 Apply this theorem to the unit ball in  $\mathbb{R}^3$  given by  $x^2 + y^2 + z^2 \leq 1$  and the starship given in Figure 1.5.

**Cylindrical algebraic decomposition.** In practice, more refined decompositions of semi-algebraic sets are used that also are computable by more or less efficient algorithms. One such decomposition is given by the *cell decomposition theorem* for semi-algebraic sets. Before we can state this theorem, we will need the notion of cylindrical algebraic decomposition (CAD) of  $\mathbb{R}^n$ : A CAD of  $\mathbb{R}^n$  is a special partition of  $\mathbb{R}^n$  into finitely many cells. The definition is by induction on  $n$ :

- (i) a CAD of  $\mathbb{R}^1$  is a collection

$$\{(-\infty, a_1), (a_1, a_2), \dots, (a_k, +\infty), \{a_1\}, \dots, \{a_k\}\},$$

of open intervals and points, where  $a_1 < \dots < a_k$  are points in  $\mathbb{R}$ .

- (ii) a CAD of  $\mathbb{R}^{n+1}$  is a finite partition of  $\mathbb{R}^{n+1}$  into (semi-algebraic) *cells*  $A$  such that the set of projections  $\pi(A)$  is again a CAD of  $\mathbb{R}^n$ . Here,  $\pi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$  is again the usual projection map defined by  $\pi(x_1, \dots, x_n, x_{n+1}) = (x_1, \dots, x_n)$ .

We still have to specify what a cell in  $\mathbb{R}^{n+1}$  is. Let  $(i_1, \dots, i_m)$  be a sequence of zeros and ones of length  $m$ . We define a cell inductively on  $m$  as follows:

- (i) a (0)-cell is a one-element set  $\{r\}$  of  $\mathbb{R}$ , a (1)-cell is an open interval  $(a, b) \subseteq \mathbb{R}$ .
- (ii) Suppose  $(i_1, \dots, i_m)$ -cells are already defined. Then an  $(i_1, \dots, i_m, 0)$ -cell is the graph  $\Gamma(f)$  of a continuous semi-algebraic function  $f : X \rightarrow \mathbb{R}$ , where  $X$  is an  $(i_1, \dots, i_m)$ -cell. Furthermore, an  $(i_1, \dots, i_m, 1)$ -cell is a set of the form

$$(f, g)_X = \{(\vec{x}, r) \in X \times \mathbb{R} \mid \vec{x} \in X \text{ and } f(\vec{x}) < r < g(\vec{x})\},$$

where  $X$  is an  $(i_1, \dots, i_m)$ -cell and  $f, g$  are continuous semi-algebraic functions on  $X$ , possibly equal to the constant functions  $+\infty$  or  $-\infty$ .

A cell in  $\mathbb{R}^n$  is an  $(i_1, \dots, i_n)$ -cell for some sequence  $(i_1, \dots, i_n)$ . A semi-algebraic  $A$  of  $\mathbb{R}^n$  is said to be *partitioned* by a CAD  $\mathcal{D}$  of  $\mathbb{R}^n$  if each cell in  $\mathcal{D}$  is either part of or disjoint with  $A$ . In other words,  $A$  is the union of cells in  $\mathcal{D}$ .

**THEOREM 1.23 (FINITE CELL DECOMPOSITION)** *Given any semi-algebraic sets  $A_1, \dots, A_k$  of  $\mathbb{R}^n$ , there is a CAD of  $\mathbb{R}^n$  partitioning each  $A_1, \dots, A_k$ .* QED

**EXAMPLE 1.24** Consider the semi-algebraic subset of  $\mathbb{R}^2$  given by the formula

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

This set is shown in part (a) of Figure ???. In part (b) of this figure, a CAD of  $\mathbb{R}^2$  is given, consisting of 25 cells, which partitions  $A$ . This CAD induces a CAD on the  $x$ -axis consisting of three (0)-cells and four (1)-cells (two of which are unbounded). On top of these intervals (0, 0), (0, 1), (1, 0), and (1, 1)-cells are built.

A proof of the finite cell decomposition theorem is given in (?, Chapter 3, Theorem 2.11). A key ingredient in this proof is the so-called *uniform finiteness property* of semi-algebraic sets. This property is also useful to obtain inexpressibility results, as will be shown in Section ??. To state this property, we need some definitions. A set  $A$  of  $\mathbb{R}^{n+1}$  is called *finite over  $\mathbb{R}^n$*  if for each  $\vec{x} \in \mathbb{R}^n$  the fiber  $A_{\vec{x}} = \{r \in \mathbb{R} \mid (\vec{x}, r) \in A\}$  is

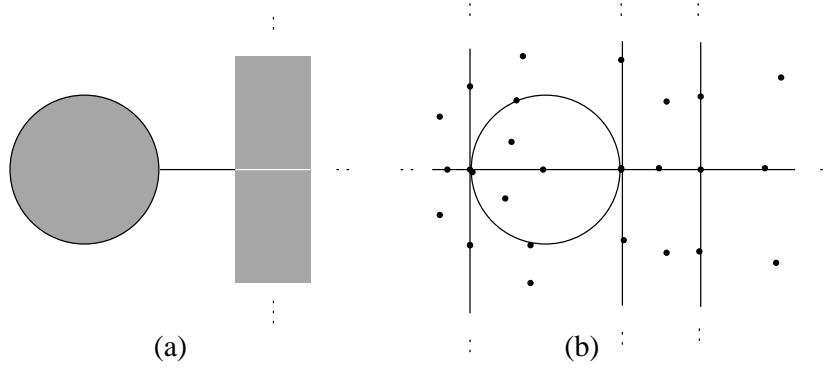


Figure 1.8. An example of a CAD in  $\mathbb{R}^2$ .

finite. We call  $A$  *uniformly finite over*  $\mathbb{R}^n$  if there is an  $N \in \mathbb{N}$  such that  $|A_{\vec{x}}| \leq N$  for all  $\vec{x} \in \mathbb{R}^n$ . We then have:

**THEOREM 1.25 (UNIFORM FINITENESS PROPERTY)** *If  $A \subseteq \mathbb{R}^{n+1}$  is a semi-algebraic set which is finite over  $\mathbb{R}^n$ , then  $A$  is uniformly finite over  $\mathbb{R}^n$ .*

As we will see in the next section, CAD is the basic tool for eliminating quantifiers. To be correct, we need an adaptation of the CAD to a given set of polynomials such that the sign of each of these polynomials is constant on each cell in the CAD. Moreover, in the context of quantifier elimination, CAD algorithms typically produce sample points in each cell, which enable to determine the sign of the polynomials. In Figure ??, we have indicated a sample point for each cell in the CAD.

**Triviality.** We remark that until now, all results hold when we replace semi-algebraic by semi-linear. However, for the following result to be true one needs to work in the semi-algebraic setting.

**EXAMPLE 1.26** Consider again the semi-algebraic set  $A$  and CAD of  $\mathbb{R}^2$  of Example ??, shown in Figure ??. Going from left to right, let  $C_1 = (-\infty, a_1)$ ,  $C_2 = \{a_1\}$ ,  $C_3 = (a_1, a_2)$ ,  $C_4 = \{a_2\}$ ,  $C_5 = (a_2, a_3)$ ,  $C_6 = \{a_3\}$  and  $C_7 = (a_3, +\infty)$  be the (0) and (1)-cells on the  $x$ -axis. If one looks at the intersections of the cylinders  $C_i \times \mathbb{R}$  with  $A$ , then it is clear that  $A \cap (C_i \times \mathbb{R})$  is semi-algebraically homeomorphic to a product  $C_i \times F_i$ , where  $F_i$  is a semi-algebraic subset of  $\mathbb{R}$ . In this example,  $F_1 = F_6 = \emptyset$ ,  $F_2 = F_4 = F_5 = \{b_1\} \in \mathbb{R}$ ,  $F_3 = [b_2, b_3] \subset \mathbb{R}$ , and  $F_7 = \mathbb{R} \setminus \{b_4\}$ . In other words, the  $x$ -axis is decomposed into cells, such that  $A$  looks like a constant set above any two points in the same cell.

One then says that the projection map  $\pi : \mathbb{R}^2 \rightarrow \mathbb{R}$  on the  $x$ -axis is *trivial* over each of the cells  $C_1, \dots, C_7$ .

We now formalize the intuition behind the example above. Let  $A \subseteq \mathbb{R}^m$  and  $B \subseteq \mathbb{R}^n$  be two semi-algebraic sets and let  $f : A \rightarrow B$  be a continuous semi-algebraic map (not necessarily a function). We can see  $A$  as a family of sets (i.e., fibers)  $\{f^{-1}(\vec{b}) \mid \vec{b} \in B\}$ . A semi-algebraic *trivialization* of  $f$  is a pair  $(F, \lambda)$  consisting of semi-algebraic set  $F \subseteq \mathbb{R}^N$ , for some  $N$ , and semi-algebraic map  $\lambda : A \rightarrow F$  such that  $(f, \lambda) : A \rightarrow B \times F$  is a homeomorphism.

**EXERCISE 1.27** Let  $f : A \rightarrow B$  and suppose that  $f$  has a semi-algebraic trivialization. Show that all fibres are semi-algebraic homeomorphic to each other.

We call  $f$  *semi-algebraically trivial* if  $f$  has a semi-algebraic trivialization. Moreover, given  $B' \subseteq B$ , we say that  $f$  is *semi-algebraically trivial over  $B'$*  if the restriction of  $f$  to  $f^{-1}(B')$  is semi-algebraically trivial.

**THEOREM 1.28 (TRIVIALITY THEOREM)** *Let  $f : A \rightarrow B$  be a continuous semi-algebraic map as above. Then there is a finite partition of  $B = B_1 \cup \dots \cup B_\ell$  such that each  $B_i$  is semi-algebraic and  $f$  is semi-algebraically trivial over each  $B_i$ .*

**EXERCISE 1.29** Consider the example of the scene from Star Trek as given in Example 1.14. Denote by  $\text{USS}_t$  the snapshot of the movie at time  $t$ . Hence, the full scene corresponds to the semi-algebraic subset  $A = \{(x, y, t) \in \text{USS}_t \mid t \in [0, 2]\}$  of  $\mathbb{R}^3$ . Let  $\pi : A \rightarrow \mathbb{R}$  be the projection on the time axis, i.e.,  $\pi(x, y, t) = t$ . The Triviality Theorem now states that we can decompose the time axis such  $\pi$  is trivial over each part in the decomposition. Find such decomposition of the time axis. What does it tell you about the scene from the Star Trek movie?

### 3.3 The local conical structure of semi-algebraic sets

Let  $A$  be a semi-algebraic set of  $\mathbb{R}^n$  and  $\vec{p}$  a point of the closure of  $A$ . Let  $B^n(\vec{p}, \varepsilon)$  be the closed ball with center  $\vec{p}$  and radius  $\varepsilon$  and let  $S^{n-1}(\vec{p}, \varepsilon)$  be the sphere with center  $\vec{p}$  and radius  $\varepsilon$ .

We denote by  $\text{Cone}(\vec{p}, S^{n-1}(\vec{p}, \varepsilon) \cap A)$  the cone with vertex  $\vec{p}$  and base  $S^{n-1}(\vec{p}, \varepsilon) \cap A$ , i.e., the set of points in  $\mathbb{R}^n$  defined by  $\lambda\vec{p} + (1 - \lambda)\vec{x}$  with  $\lambda \in [0, 1]$  and  $\vec{x} \in S^{n-1}(\vec{p}, \varepsilon) \cap A$ . Let  $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$  denote the standard Euclidean norm.



**THEOREM 1.30 (LOCAL CONICAL STRUCTURE)** *For any point  $p$  of the closure of a semi-algebraic set  $A$ , there exists a number  $\varepsilon > 0$  and a semi-algebraic homeomorphism*

$$h : B^n(\vec{p}, \varepsilon) \cap A \rightarrow \text{Cone}(\vec{p}, S^{n-1}(\vec{p}, \varepsilon) \cap A)$$

*such that  $\|h(\vec{x}) - \vec{p}\| = \|\vec{x} - \vec{p}\|$  and  $h|_{S^{n-1}(\vec{p}, \varepsilon) \cap A} = \text{Id}$ .*

A radius  $\varepsilon > 0$  given by the previous theorem is called a cone radius of  $A$  in  $\vec{p}$ .

**EXERCISE 1.31** Prove the local conical structure theorem using the triviality theorem. (Hint. Take for  $f : A \rightarrow \mathbb{R}$  the mapping  $f(\vec{x}) = \|\vec{x} - \vec{p}\|$ .)

We examine the local conical structure of semi-algebraic sets of  $\mathbb{R}^2$  in more detail. Consider the semi-algebraic set of  $\mathbb{R}^2$  depicted in Figure ?? . For the point  $\vec{p}$ , we have indicated a cone radius  $\varepsilon$  of  $A$  in  $\vec{p}$  by the dotted lines. The intersection  $S^1(\vec{p}, \varepsilon) \cap A$  consists of a finite number of points and open intervals. If we denote open intervals that belong  $A$  by  $R^+$  and intervals that belong to the complement of  $A$  by  $R^-$ , and if we similarly indicate points belonging to  $A$  by  $L^+$  and points belonging to the complement by  $L^-$ , we can describe a the intersection  $S^1(\vec{p}, \varepsilon) \cap A$  by means of a circular list over the alphabet  $\{L^+, L^-, R^+, R^-\}$ . We call this circular list the *cone type of  $A$  in  $\vec{p}$* . The symbols  $L$  and  $R$  refer to lines and regions that arrive at  $p$ . There are two exceptions, however. For a point in the topological interior of  $A$ , we have that  $S^1(\vec{p}, \varepsilon) = S^1(\vec{p}, \varepsilon) \cap A$ , which we denote by  $F$  (for full). On the other hand, for an isolated point of  $A$ , we have  $S^1(\vec{p}, \varepsilon) \cap A = \emptyset$ , which we denote by  $E$  (for empty). For instance, the cone type of  $A$  in  $\vec{p}$  in Figure ?? is given by

$$(L^+ R^- L^+ R^- L^- R^+ L^+ R^- L^+ R^- L^+ R^+ L^- R^+).$$

A semi-algebraic set  $A$  of  $\mathbb{R}^2$  also has a local conical structure at infinity. To see this, we embed  $\mathbb{R}^2$  as the  $(x, y)$ -plane in  $\mathbb{R}^3$  and map  $A$  from this embedded plane onto the sphere  $S^2((0, 0, 1), 1)$ , that rests on the  $(x, y)$ -plane, in the direction of its north pole  $(0, 0, 2)$ . If we then add the north pole to this set as the point at infinity of the semi-algebraic set, rotate the sphere such that  $(0, 0, 2)$  becomes the origin, and stereographically project back on the  $xy$ -plane, then the local conical structure of  $(0, 0)$  in the resulting semi-algebraic set reveals the conical structure of the point at infinity in  $A$ .

This implies that there exists a  $\varepsilon > 0$  such that  $\{(x, y) \mid x^2 + y^2 \geq \varepsilon^2\} \cap A$  is homeomorphic to  $\{(\lambda x, \lambda y) \mid (x, y) \in S^1((0, 0), \varepsilon) \cap A \wedge \lambda \geq 1\}$ . We can indeed view the latter set as the cone with top  $\infty$  and base  $S^1((0, 0), \varepsilon) \cap A$ .

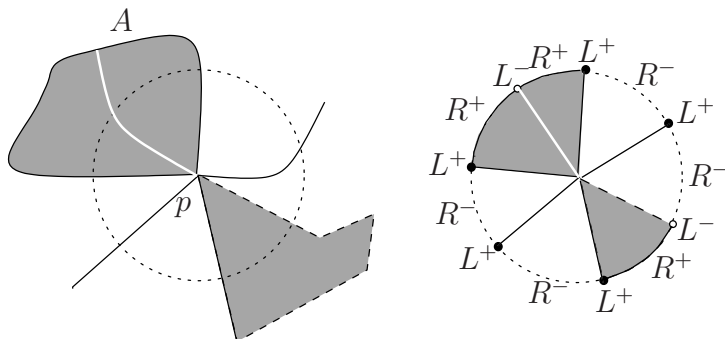


Figure 1.9. A semi-algebraic set  $A$  of  $\mathbb{R}^2$  and the cone type of  $A$  in its points  $p$  given by the circular list  $(L^+R^-L^+R^-L^-R^+L^+R^-L^+R^-L^+R^+L^-R^+)$ .

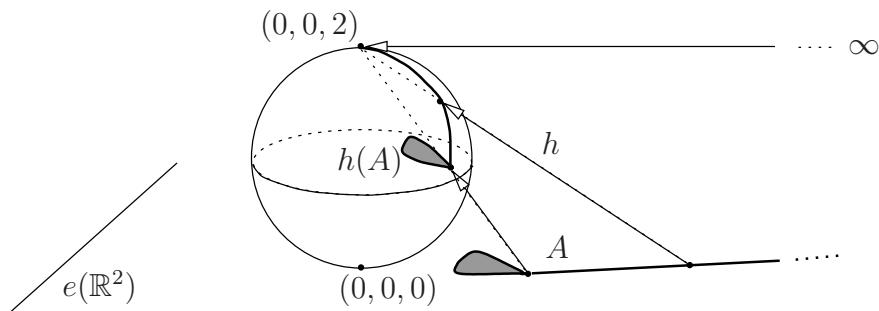


Figure 1.10. Illustration of the stereographical projection  $h$ .

More formally, consider the embedding  $e$  of  $\mathbb{R}^2$  in  $\mathbb{R}^3$  that maps  $(x, y)$  to  $(x, y, 0)$ . Let  $\sigma$  be the reflection of  $\mathbb{R}^3$  defined by  $(x, y, z) \mapsto (x, y, 2 - z)$ . Finally, let  $h : e(\mathbb{R}^2) \cup \{\infty\} \rightarrow S^2((0, 0, 1), 1)$  be the homeomorphism of that maps the Alexandrov one-point compactification of  $e(\mathbb{R}^2)$  stereographically onto the sphere  $S^2((0, 0, 1), 1)$ , i.e.,  $h(x, y, 0) = \frac{4}{4+x^2+y^2}(x, y, \frac{x^2+y^2}{2})$  and  $h(\infty) = (0, 0, 2)$ .

We define the cone type of  $A$  in  $\infty$  to be the cone type of the point  $(0, 0)$  in the set  $e^{-1}(h^{-1}(\sigma(\{(0, 0, 2)\} \cup h(e(A))) \setminus \{\infty\}))$ . Remark that the cone type of  $A$  in  $\infty$  is  $(E)$  if and only if  $A$  is a bounded subset of  $\mathbb{R}^2$ .

**EXERCISE 1.32** Let  $A$  be a semi-algebraic set in  $\mathbb{R}^2$  and let  $\vec{p}$  be a point in the closure of  $A$ .

- 1 Verify that  $e^{-1}(h^{-1}(\sigma(\{(0, 0, 2)\} \cup h(e(A))) \setminus \{\infty\}))$  is indeed a semi-algebraic set.
- 2 The notion of the cone type of  $A$  in  $\vec{p}$  is well-defined.

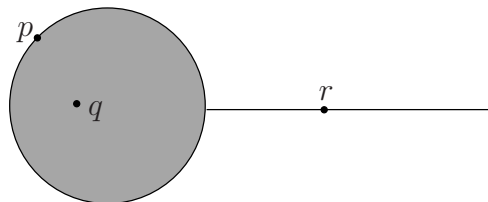


Figure 1.11. Types of regular points of a closed semi-algebraic set:  $\Pi(p) = (R)$ ,  $\Pi(q) = F$  and  $\Pi(r) = (LL)$ .

3 The notion of the cone type of  $A$  in  $\infty$  is well-defined.

(Hint: For 2 and 3, prove that for any two cone radii  $\varepsilon_1$  and  $\varepsilon_2$  of  $A$  in  $\vec{p}$  (and  $\infty$ ) we get the same cone type.)

Let  $\mathcal{C}$  be the set of all possible cone types of semi-algebraic sets of  $\mathbb{R}^2$ . We define:

DEFINITION 1.33 Let  $A$  be a semi-algebraic set of  $\mathbb{R}^2$ . The *point-structure* of  $A$  is the function  $\Pi(A)$  from  $A \cup \{\infty\}$  to  $\mathcal{C}$  that maps each point in the closure of  $A$  to its cone type.

An important observation is the following:

PROPOSITION 1.34 *The number of points in the closure of  $A$  with a cone different from  $(R^-L^-R^+L^-)$ ,  $(R^-L^+R^+L^+)$ ,  $(R^-L^+R^-L^+)$ ,  $(R^+L^-R^+L^-)$  and  $F$  is finite.*

We call point in the closure of  $A$  *singular* if it has a cone type different from  $(R^-L^-R^+L^-)$ ,  $(R^-L^+R^+L^+)$ ,  $(R^-L^+R^-L^+)$ ,  $(R^+L^-R^+L^-)$  or  $F$ . Otherwise, we say that a point in the closure of  $A$  is *regular*.

EXERCISE 1.35 Let  $A$  be a semi-algebraic set of  $\mathbb{R}^2$ .

- 1 Prove Proposition ??.
- 2 Show that only a finite number of cone types can appear in  $A$ .
- 3 If  $A$  contains a point of cone type  $(R^-L^-R^+L^-)$ ,  $(R^-L^+R^+L^+)$ ,  $(R^-L^+R^-L^+)$ ,  $(R^+L^-R^+L^-)$  and  $F$ , then it must have infinitely many points of this cone type.

EXAMPLE 1.36 Suppose that  $A$  is a closed semi-algebraic set of  $\mathbb{R}^2$ . By the previous exercise, there are infinitely many points in which  $A$  has one of the following three cone types  $(R^-L^+R^+L^+)$ ,  $(R^-L^+R^-L^+)$ , and  $(F)$ . Figure ?? illustrates these different cone types.

In Section ??, we will show the importance of the cone types and point structure for the expressibility of first-order logic over the reals.

### 3.4 Triangulations

An interesting question is whether semi-algebraic sets can exhibit more topological properties than semi-linear sets. The following results shows that this is not the case. Roughly speaking, the *triangulation theorem* states that each semi-algebraic set is homeomorphic to a semi-linear one. To make this more precise, we need the following notations.

Let  $a_0, a_1, \dots, a_k$  be  $(k + 1)$  affine independent points in  $\mathbb{R}^n$ . A *k-simplex*  $(a_0, a_1, \dots, a_k)$  is the set of points

$$(a_0, a_1, \dots, a_k) = \left\{ \sum t_i a_i \mid \text{all } t_i > 0, \sum t_i = 1 \right\} \subseteq \mathbb{R}^n.$$

Note that *k-simplex* is of dimension *k*. Let  $\sigma$  be a *k-simplex* given by  $(a_0, a_1, \dots, a_k)$ . The *closure of  $\sigma$* , denoted by  $\text{cl}(\sigma)$  is the set of points

$$\text{cl}(\sigma) = \left\{ \sum t_i a_i \mid \text{all } t_i \geq 0, \sum t_i = 1 \right\}.$$

A *face of  $\sigma$*  is a simplex corresponding to any nonempty subset of  $(a_0, a_1, \dots, a_k)$ . A *complex in  $\mathbb{R}^n$*  is a finite collection  $K$  of simplices in  $\mathbb{R}^n$ , such that for all  $\sigma_1, \sigma_2 \in K$ , either  $\text{cl}(\sigma_1) \cap \text{cl}(\sigma_2) = \emptyset$ , or  $\text{cl}(\sigma_1) \cap \text{cl}(\sigma_2) = \text{cl}(\tau)$ , where  $\tau$  is a common face of  $\sigma_1$  and  $\sigma_2$ . We denote by  $|K|$  the union of the simplices of  $K$ . From the definition it is clear that  $|K|$  is a bounded semi-linear set of  $\mathbb{R}^n$ .

**THEOREM 1.37** *Let  $A$  be a semi-algebraic set of  $\mathbb{R}^n$ . Then there exists a complex  $K$  in  $\mathbb{R}^n$  and a semi-algebraic homeomorphism  $h$  such that  $h(A) = |K|$ , i.e.,  $A$  is semi-algebraically homeomorphic to  $|K|$ .*

## 4. Query evaluation through quantifier elimination

In this section, we address in more detail how queries, expressible in the logics  $\text{FO}(+, <, 0, 1)$  and  $\text{FO}(+, \times, <, 0, 1)$ , may be evaluated.

When we have a query expressed by a formula  $\varphi(x_1, \dots, x_m)$  over the vocabulary  $(+, \times, <, 0, 1, S_1, \dots, S_k)$  and we want to evaluate this query on a concrete input database over the schema  $\mathcal{S} = (S_1, \dots, S_k)$ , given by quantifier-free formulas  $\varphi_{S_1}(x_1, \dots, x_{n_1}), \dots, \varphi_{S_k}(x_1, \dots, x_{n_k})$  in  $\text{FO}(+, \times, <, 0, 1)$  ( $n_i$  is the arity of  $S_i$ ,  $i = 1, \dots, k$ ), we can proceed as follows:

- we plug-in the descriptions  $\varphi_{S_1}(x_1, \dots, x_{n_1}), \dots, \varphi_{S_k}(x_1, \dots, x_{n_k})$  of the input relations into the query formula  $\varphi(x_1, \dots, x_m)$  (this means that we replace each occurrence of some  $S_i(v_1, \dots, v_{n_i})$  in the query formula by  $\varphi_{S_i}(v_1, \dots, v_{n_i})$ );

- this results in a formula over the vocabulary  $(+, \times, <, 0, 1)$  that may contain quantifiers introduced by the query formula;
- next, we eliminate these quantifiers and obtain a quantifier-free description in  $\text{FO}(+, \times, <, 0, 1)$  of the output relation.

We remark that the same query evaluation strategy may be applied when  $\times$  is omitted.

EXAMPLE 1.38 The formula

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

over the schema  $(+, \times, <, 0, 1)$  expresses the topological interior of a set  $S$  in  $\mathbb{R}^2$ . When we want to evaluate the query expressed by this formula on the disk given by  $x^2 + y^2 \leq 4$ , we first replace  $S(x', y')$  in the query formula by  $(x')^2 + (y')^2 < 4$ . This gives rise to the formula

$$\psi(x, y) = \exists \varepsilon (\varepsilon \neq 0 \wedge \forall x' \forall y' ((x - x')^2 + (y - y')^2 < \varepsilon^2 \rightarrow (x')^2 + (y')^2 < 4)).$$

The formula  $\psi$  contains three quantifiers. When we eliminate the quantifiers from  $\psi$ , we obtain as canonical quantifier-free description of the output the formula  $x^2 + y^2 < 4$ .

The reader might wonder, why we bother about eliminating quantifiers. Indeed, simply plugging in the  $\text{FO}(+, \times, <, 0, 1)$ -formulas of the input relations into the query formula, yields a formula in  $\text{FO}(+, \times, <, 0, 1)$  that also describes the output. And these formulas, even though containing quantifiers, may be used in turn to describe an input to further queries. Even without eliminating quantifiers, we would have a formalism that has this *closure property*. Closure is a much desired property in database theory where it is considered important that outputs of queries may serve as input for further queries (compositionality of queries).

So, why is it so relevant to eliminate quantifiers? The answer lies in the question of what can we do with these formulas that describe output relations. Or rather, we should ask what we would like to do with these defining formulas.

Typical questions that are asked in database practice are the following:

- *Membership test*: for example, does  $(1, 2)$  belong to the output relation given by  $\varphi(u, v) = \exists x \exists y (u = x + y \wedge ((x = 1 \vee x = 2) \wedge y = 3)) \vee u = v$ ?
- *Emptiness test*: for example, is the set  $S$  given by  $\exists x \exists y (z = x + y \wedge ((x = 1 \vee x = 2) \wedge y = 3))$  empty?

We observe that both questions, which are relevant to database practice, add up to deciding the truth of sentences of  $\text{FO}(+, \times, <, 0, 1)$ . Indeed, to answer the membership test the truth of the sentence

$$\exists x \exists y (1 = x + y \wedge ((x = 1 \vee x = 2) \wedge y = 3)) \vee 1 = 2$$

has to be decided. For the second test, the truth of the sentence

$$\exists z \exists x \exists y (z = x + y \wedge ((x = 1 \vee x = 2) \wedge y = 3))$$

has to be determined.

Deciding the truth of sentences is possible in decidable theories like  $(\mathbb{R}, +, <, 0, 1)$  and  $(\mathbb{R}, +, \times, <, 0, 1)$ . We discuss decision procedures for these theories in the subsections that follow.

#### 4.1 Quantifier elimination for $\text{FO}(+, <, 0, 1)$

The theory of  $(\mathbb{R}, +, <, 0, 1)$  has the following quantifier elimination property.

**THEOREM 1.39** *The theory of  $(\mathbb{R}, +, <, 0, 1)$  admits quantifier elimination. More specifically, this means that there is an algorithm that on input a formula  $\varphi(x_1, \dots, x_n)$  over  $(+, <, 0, 1)$  returns a quantifier-free formula  $\psi(x_1, \dots, x_n)$  that is equivalent to  $\varphi(x_1, \dots, x_n)$  over  $(\mathbb{R}, +, <, 0, 1)$ .*

We say that two formulas  $\varphi(x_1, \dots, x_n)$  and  $\psi(x_1, \dots, x_n)$  are *equivalent* if they define the same  $n$ -ary relation over  $\mathbb{R}$ .

For  $(\mathbb{R}, +, <, 0, 1)$ , there turns out to be a conceptually very simple quantifier elimination procedure that goes back to Fourier in 1826 and that was rediscovered by Motzkin in 1936 (?) and by several other researchers, even as late as the second half of the 20th century. We sketch the algorithm of Fourier now. Let us concentrate on the problem of eliminating a single existential quantifier from a formula  $\varphi(x_1, \dots, x_{m-1})$  of the form

$$\exists x_m \psi(x_1, \dots, x_m),$$

where  $\psi(x_1, \dots, x_m)$  is a Boolean combination of atomic formulas of  $\text{FO}(+, <, 0, 1)$ . So,  $\psi(x_1, \dots, x_m)$  can be written as

$$\bigvee_{i=1}^d \bigwedge_{j=1}^{e_i} x_m \theta_{ij} c_{0ij} + \sum_{k=1}^{m-1} c_{kij} x_i$$

with  $\theta_{ij} \in \{=, <, \leq, >, \geq\}$ . If we abbreviate the terms  $c_{0ij} + \sum_{k=1}^{m-1} c_{kij} x_i$  by  $t_{ij}$ , then we can remark that for any values given to the variables  $x_1, \dots, x_{m-1}$ , the terms  $t_{ij}$  can be ordered, let us say (after re-indexing)

as  $t_1 \leq t_2 \leq \dots \leq t_k$ . It is clear that for any two values of  $x_m$  taken strictly between some  $t_i$  and  $t_{i+1}$  (see Figure ??), the truth value of  $x_m \theta_{ij} c_{0ij} + \sum_{k=1}^{m-1} c_{kij} x_i$  is the same. The same is true if we take any two values of  $x_m$  strictly smaller than  $t_1$  or strictly larger than  $t_k$ .

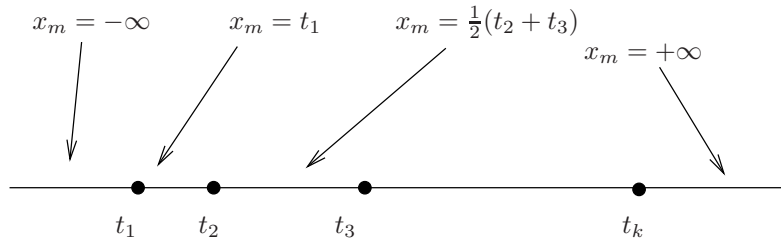


Figure 1.12. The relevant values for  $x_m$ .

Therefore, when the  $x_1, \dots, x_{m-1}$  vary, the existential quantifier in  $\exists x_m \psi(x_1, \dots, x_m)$  is equivalent expressible by the disjunction

$$\bigvee_{x_m = t_i \text{ OR } x_m = 1/2(t_i + t_j) \text{ OR } x_m = \pm\infty} \psi(x_1, \dots, x_m).$$

In this disjunction, for  $x_m$  all values on and in between all of the  $t_{ij}$  are considered. In the above formula  $x_m = \pm\infty$  can be achieved by considering all values  $t_i \pm 1$  for  $x_m$ . It is clear that the above given disjunction suffices to replace the quantifier.

We remark that this procedure takes exponential space in the size of the input formula and that the time complexity of this procedure is doubly exponential.

EXAMPLE 1.40 Suppose we want to eliminate the quantifier in

$$\varphi(x, y) = \exists z(x + z = y \wedge z < 2 + x).$$

To start with, we write this formula in the right form, namely  $\exists z(z = y - x \wedge z < 2 + x)$ . When we apply the procedure described above, we get the quantifier-free formula  $\psi(x, y) =$

$$\begin{aligned}
& (y - x = y - x \wedge y - x < 2 + x) \vee \\
& (2 + x = y - x \wedge 2 + x < 2 + x) \vee \\
& (y - x - 1 = y - x \wedge y - x - 1 < 2 + x) \vee \\
& (2 + x - 1 = y - x \wedge 2 + x - 1 < 2 + x) \vee \\
& (y - x + 1 = y - x \wedge y - x + 1 < 2 + x) \vee \\
& (2 + x + 1 = y - x \wedge 2 + x + 1 < 2 + x) \vee \\
& \left( \frac{y - x + 2 + x}{2} = y - x \wedge \frac{y - x + 2 + x}{2} < 2 + x \right).
\end{aligned}$$

As mentioned above, we remark that the  $-\infty$  and  $+\infty$  from the algorithm are implemented by subtracting and adding 1 from the terms  $t_1 = y - x$  and  $t_2 = 2 + x$  respectively. Also remark that many atomic formulas in this expression for  $\psi(x, y)$  are trivially true or false. So,  $\psi(x, y)$  could be further simplified.

**EXERCISE 1.41** Evaluate the sentence from Exercise 1.13 that expresses that a binary linear input relation is convex on the input given by  $x = y \wedge x > 0$  by plugging in this description in the query formula and eliminating the quantifiers.

## 4.2 Quantifier elimination for $\text{FO}(+, \times, <, 0, 1)$

In the 1930s, Alfred Tarski showed that  $\text{FO}(+, \times, <, 0, 1)$  has the algorithmic quantifier elimination property too. Tarski published this result only in 1948 (?).

**THEOREM 1.42** *There is an algorithm that on input a  $\text{FO}(+, \times, <, 0, 1)$ -formula  $\varphi(x_1, \dots, x_n)$  returns a quantifier-free  $\text{FO}(+, \times, <, 0, 1)$ -formula  $\psi(x_1, \dots, x_n)$  that is equivalent to the given formula  $\varphi(x_1, \dots, x_n)$  over  $(\mathbb{R}, +, \times, <, 0, 1)$ .*

The quantifier elimination procedure given by Tarski is based on a theorem by Sturm on real root counting and has a huge complexity (it is not elementary recursive), which makes it unsuitable for practical purposes.

**EXAMPLE 1.43** A well-known example of quantifier elimination is the following. Consider the formula

$$\varphi(a, b, c) = a \neq 0 \wedge \exists x(ax^2 + bx + c = 0).$$

This formula describes triples  $(a, b, c)$  for which the quadratic equation  $ax^2 + bx + c = 0$  has a real root. From high-school mathematics, we



know that  $\varphi(a, b, c)$  is equivalent to the quantifier-free formula

$$\psi(a, b, c) = a \neq 0 \wedge (b^2 - 4ac \geq 0).$$

Improvements to Tarski's procedure were proposed by Seidenberg (?), but a major breakthrough was achieved by George Collins (?) in 1975, when he introduced the *cylindrical algebraic decomposition* (CAD) of semi-algebraic sets. His algorithm takes as input a system of polynomial equalities and inequalities that describe a semi-algebraic set in some  $\mathbb{R}^n$ . The algorithm returns a partitioning of  $\mathbb{R}^n$  in a finite number of cells that are described by sign conditions on polynomials in  $n$  variables. These cells are actually accompanied by sample points in each of the cells that allow to determine the sign conditions of these polynomials in these cells. The algorithm of Collins to compute a CAD has in the worst-case doubly-exponential sequential time complexity in the number of variables. It was the first quantifier elimination algorithm that has been implemented, however. It has undergone numerous improvements, resulting in the implementation QEPCAD (Quantifier Elimination by Partial Cylindrical Algebraic Decomposition) by Hong (?). We refer to (?) for a description at length of the current state of CAD.

A formal definition of a CAD was given in Section 1.3. It is beyond the scope of this chapter to give a full description of Collins' CAD algorithm, but we want to give an idea of the major steps in the algorithm.

Suppose the input of the CAD algorithm is an  $\text{FO}(+, \times, <, 0, 1)$ -formula in prenex normal form

$$\varphi(u_1, \dots, u_m) = \exists x_1 \cdots \exists x_n \bar{\varphi}(u_1, \dots, u_m, x_1, \dots, x_n).$$

Here,  $\bar{\varphi}(u_1, \dots, u_m, x_1, \dots, x_n)$  is a boolean combination of expressions of the form  $p = 0$ ,  $p > 0$ ,  $p \geq 0$  or  $p \neq 0$ , where  $p$  is a polynomial. It is custom in the quantifier elimination literature to distinguish between the variables  $(x_1, \dots, x_n)$  and parameters  $(u_1, \dots, u_m)$  of the given formula  $\varphi(u_1, \dots, u_m, x_1, \dots, x_n)$ . The goal is to eliminate the variables from the formula  $\varphi(u_1, \dots, u_m)$  via the computation of a CAD of  $\bar{\varphi}(u_1, \dots, u_m, x_1, \dots, x_n)$ .

The main construction steps in the construction of a CAD of the set  $A = \{(u_1, \dots, u_m, x_1, \dots, x_n) \in \mathbb{R}^{m+n} \mid \bar{\varphi}(u_1, \dots, u_m, x_1, \dots, x_n)\}$  are:

- *the projection phase:* here the  $(m+n)$ -dimensional semi-algebraic set  $A$  is iteratively projected onto lower dimensional spaces ( $\mathbb{R}^{m+n} \rightarrow \mathbb{R}^{m+n-1} \rightarrow \dots \rightarrow \mathbb{R}^1$ );
- *the basis phase:* here real roots are isolated in  $\mathbb{R}^1$  and sample points are computed (using numeric methods);

- *the extension phase:* here, again in an iterative way ( $\mathbb{R}^1 \rightarrow \mathbb{R}^2 \rightarrow \dots \rightarrow \mathbb{R}^{m+n-1} \rightarrow \mathbb{R}^{m+n}$ ), a lifting to higher dimensions takes place. Stacks of cells (sections and sectors) are built, iteratively, together with sample points.

The output of the CAD algorithm is a sequence  $C_1, \dots, C_{m+n}$ , where each  $C_i$  is a partition of  $\mathbb{R}^i$ . These cells are given by means of quantifier-free FO(+, ×, <, 0, 1)-formulas and a sample point. In particular, for each of the cells in the resulting decomposition of  $\mathbb{R}^{m+n}$  it is recorded whether it belongs to the given semi-algebraic set  $A$  or not.

A quantifier-free equivalent formula for  $\varphi(u_1, \dots, u_m)$  is then obtained as a disjunction of all formulas  $\varphi_C$  describing cells  $C$  in  $C_m$  such that in the stack above  $C$  a cell of  $C_{m+n}$  belongs to  $A$ .

EXAMPLE 1.44 We illustrate the CAD algorithm using the three-dimensional set given by the quantifier-free FO(+, ×, <, 0, 1)-formula

$$x^2 + y^2 + z^2 \leq 1 \vee (x^2 + y^2 + (z - 2)^2 = 1 \wedge z \leq 5/2) \vee (x^2 + y^2 + (z - 3)^2 = 1 \wedge z > 5/2).$$

This set is depicted in Figure ??.

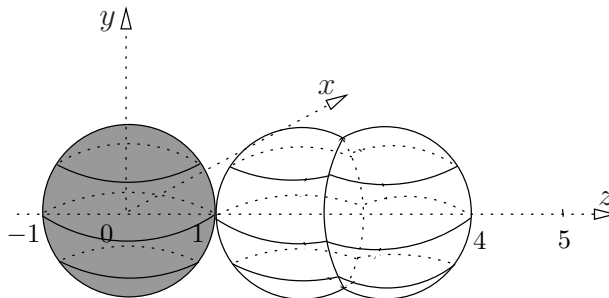


Figure 1.13. An example of a semi-algebraic set in  $\mathbb{R}^3$ .

In the projection phase this three-dimensional set is projected on  $(x, z)$ -plane and then this projected set is in turn projected on the  $z$ -axis (details omitted). On the real line certain “special points” are determined. In Figure ??, these special points are coloured grey on the  $z$ -axis. These special points are always finite in number and they partition the line  $\mathbb{R}$  into a finite number of points and open intervals (two of which are unbounded).

In the example of Figures ?? and ??, there are 7 points and 8 intervals. This partition is called the one-dimensional induced CAD of the given set. Next, in the extension phase, stacks are built on the one-dimensional

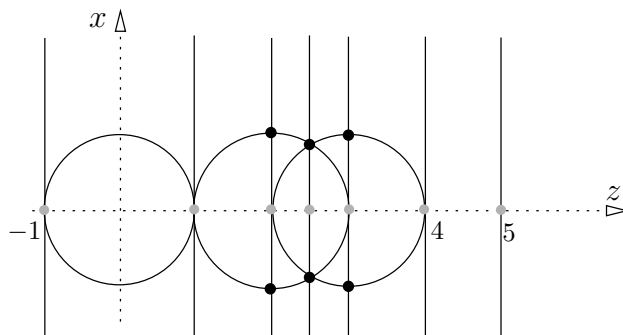


Figure 1.14. The 1- and 2-dimensional induced CADs of the semi-algebraic set of Figure ??.

CAD. The stack above the second interval, for instance, consists of two curves (called sections) and three regions (called sectors), two of which are unbounded. The cells in these stacks form the two-dimensional induced CAD of the given set. Finally, stacks are built on these cells, resulting in the CAD of the given set, or to be more precise, of the description of the given set. For each of the cells in this decomposition of  $\mathbb{R}^3$  it is recorded whether it belongs to the given semi-algebraic set or not.

During the 1990s more efficient quantifier elimination algorithms have been proposed. In 1990, Heintz, Roy, Solerno show a doubly exponential sequential time complexity in number of quantifier alternations, rather than in the number of quantifiers (?). Later on, Heintz et al. show single exponential complexity if you work with alternative data structures, such as arithmetic boolean circuits, to store systems of polynomial equalities and inequalities (?). In the TERA project, the software *Kronecker* was developed and it is for the moment the most efficient software for quantifier elimination (?) over the reals. The Kronecker implementation is described in (?). We refer to (?) for a detailed overview of algorithms in real algebraic geometry.

## 5. Expressiveness results

In this section, we discuss some results concerning the expressive power of  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  as query languages for constraint databases.

The development of constraint databases has given rise to two directions of research. Firstly, classical relational database questions have been reconsidered. For instance, it is known that *graph connectivity* of finite relations is not expressible in first-order logic over relations (the

same holds for other properties such as *parity*, *majority*, etc.). These expressiveness results can be re-addressed in the presence of arithmetical operations. Indeed, when we assume that the finite relations are embedded in the reals, we can ask whether connectivity, parity, majority, etc., are expressible when the vocabulary of first-order logic is extended with  $+$ ,  $\times$ ,  $<$ ,  $0$  and  $1$ . Secondly, expressiveness questions related to the possibility of representing infinite relations have been studied. In this section, we start by giving some results on finite relations and then show how they help to settle questions concerning the expressive power of  $\text{FO}(+, \times, <, 0, 1)$  on infinite relations.

## 5.1 Expressiveness results for finite databases

Here, we state a *generic collapse* result which allows to reduce – or collapse – expressiveness questions in the presence of arithmetic, to the arithmetic-free case (?). We illustrate its implications on the first-order expressiveness of properties over finite databases over the reals. We also give the *dichotomy theorem* which gives a bound on the query result (in case it is finite) for first-order expressible queries (?). This bound can be used to show inexpressibility results, as we shall illustrate.

Consider the following decision problems on finite relations:

- The decision problem MAJORITY about two finite sets  $S_1$  and  $S_2$  is: MAJORITY( $S_1, S_2$ ) is true if and only if  $S_2 \subseteq S_1$  and  $|S_1| \leq 2|S_2|$ ;
- The decision problem PARITY about a finite set  $S$  is: PARITY( $S$ ) is true if and only if  $|S|$  is even.

The proof of the following lemma is a routine exercise in finite-model theory (?). It can, e.g., be proven using the well-known technique of Ehrenfeucht-Fraïssé games. This lemma holds for arbitrary finite structures.

LEMMA 1.45 *On finite structures over the signature  $(<, S_1, S_2)$ , the decision problem MAJORITY( $S_1, S_2$ ) is not expressible in  $\text{FO}(<, S_1, S_2)$ . Likewise, on finite structures over the signature  $(<, S)$ , the decision problem PARITY( $S$ ) is not expressible in  $\text{FO}(<, S)$ .*

Benedikt, Dong, Libkin and Wong proved that any first-order formula over the reals that is invariant under monotone bijections from  $\mathbb{R}$  to  $\mathbb{R}$  is equivalently expressible on *finite* relations in the restriction of first-order logic that only uses order constraints (?). This collapse result was a breakthrough in the line of research towards understanding of the expressive power of first-order logic over the reals and related structures (?; ?; ?; ?; ?; ?; ?).

Consider structures over the vocabulary  $(+, \times, <, 0, 1, S_1, \dots, S_k)$  that are expansions of  $\mathbb{R}$  with  $k$  finite relations on  $\mathbb{R}$ . We call such structures *finite structures over the reals* (to emphasize the difference with finite structures in the sense of relational databases). A first-order formula over the vocabulary  $(+, \times, <, 0, 1, S_1, \dots, S_k)$  is called *order-generic* if on such structures, it is invariant under monotone bijections  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Benedikt, Dong, Libkin, and Wong showed the following (?):

**THEOREM 1.46 (COLLAPSE THEOREM)** *For each order-generic formula in  $\text{FO}(+, \times, <, 0, 1, S_1, \dots, S_k)$ , there exists a formula in  $\text{FO}(<, S_1, \dots, S_k)$ , that is equivalent to it on finite structures over the reals. Furthermore, in the latter formula the quantifiers may be assumed to range only over the constants actually occurring in the relations  $S_1, \dots, S_k$ .*

**EXERCISE 1.47** Verify that the properties PARITY and MAJORITY of finite structures over the reals are invariant under monotone bijections from  $\mathbb{R}$  to  $\mathbb{R}$ .

The following lemma, which specializes Lemma ?? from general finite ordered structures to finite structures over the reals, now follows directly from Theorem ?? and Lemma ??.

**LEMMA 1.48** *On finite structures over the signatures  $(+, \times, <, 0, 1, S_1, S_2)$ , the decision problem MAJORITY( $S_1, S_2$ ) is not first-order expressible. Similarly, on finite structures over the signatures  $(+, \times, <, 0, 1, S)$ , the decision problem PARITY( $S$ ) is not first-order expressible.*

Apart from the above collapse results, there are also other results that are handy to show that the expressive power of  $\text{FO}(+, \times, <, 0, 1)$  is rather limited on finite structures over the reals.

**THEOREM 1.49 (DICHOTOMY THEOREM)** *Let  $\varphi$  be a  $\text{FO}(+, \times, <, 0, 1, S_1, \dots, S_k)$ -formula. There exists a polynomial  $p_\varphi$  such that for any finite structure over the reals  $D$ , the query result of  $\varphi(D)$  is either infinite or bounded by  $p_\varphi(|D|)$ , where  $|D|$  denotes the number of elements in the structure  $D$ .*

**EXERCISE 1.50** Show, using the dichotomy theorem, that the query “Return all prime divisors of the elements in  $S$ ”, where  $S$  is a unary relation name (assumed to contain natural numbers), is not expressible in  $\text{FO}(+, \times, <, 0, 1, S)$ .

## 5.2 Inexpressibility results for infinite databases

We now apply the Dichotomy theorem to obtain an inexpressibility result for infinite databases. The example that we want to discuss concerns the linear  $\varepsilon$ -approximation of semi-algebraic sets.

DEFINITION 1.51 Let  $A$  be a semi-algebraic set of  $\mathbb{R}^2$ . A (linear)  $\varepsilon$ -approximation of  $A$  is a semi-linear set  $B$  of  $\mathbb{R}^2$  which is homeomorphic to  $A$  via an homeomorphism  $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ , and such that for any  $\vec{p} \in A$ ,  $d(\vec{p}, h(\vec{p})) < \varepsilon$ .

THEOREM 1.52 Let  $\varepsilon > 0$  be a real number. There is no  $\text{FO}(+, \times, <, 0, 1, S)$ -formula that expresses a linear  $\varepsilon$ -approximation of a relation  $S$  in  $\mathbb{R}^2$ .

*Proof* Consider the query  $Q$  that returns the empty set if the relation  $S$  does not consist of three non-collinear points, and otherwise returns the corner points of an  $\varepsilon$ -approximation of the circle determined by the three points of  $S$ . Here, a corner point is a point in which two straight-line segments make an angle different from 180 degrees.

Clearly, the construction of a circle through three points is expressible in  $\text{FO}(+, \times, <, 0, 1)$  (the reader may want to verify this!). The same holds for the selection of the corner points of a semi-linear set (the reader may want to verify this as an exercise too). Hence, if we assume that the  $\varepsilon$ -approximation query can be expressed  $\text{FO}(+, \times, <, 0, 1, S)$ , then  $Q$  is also expressible in  $\text{FO}(+, \times, <, 0, 1, S)$  by a formula  $\varphi$ . However, the number of corner points which is equal to  $|\varphi(D)|$ , can be made arbitrarily large by choosing  $D$  to consist of three far enough apart points. This contradicts the dichotomy theorem, which guarantees the existence of a polynomial  $p_\varphi$  such that the output of  $\varphi$ , when applied to  $D$  is bounded by  $p_\varphi(|D|) = p_\varphi(3)$ . QED

EXERCISE 1.53 Prove Theorem ?? using the uniform finiteness property of semi-algebraic sets instead of the dichotomy theorem.

### 5.3 Expressing topological properties of spatial databases

In this section, we will show that *topological connectivity* of planar geometric figures is not expressible in  $\text{FO}(+, \times, <, 0, 1)$ . First, we remark that topological connectivity of a set in the plane is a query that is invariant under topological transformations of the plane, i.e., it is invariant under isotopies  $i : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  (an isotopy is an orientation-preserving homeomorphism, i.e., it can be seen as a stretching transformation of the plane seen as a rubber sheet). Such queries are called *topological queries*. Formally, a query  $Q$  (over an input schema with one binary relation  $S$ ) is called *topological* if for any two instances  $A$  and  $B$  of  $S$  for which there is an isotopy  $i : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  such that  $i(A) = B$ ,  $i(Q(A)) = Q(B)$  holds. This concept is illustrated in Figure ??

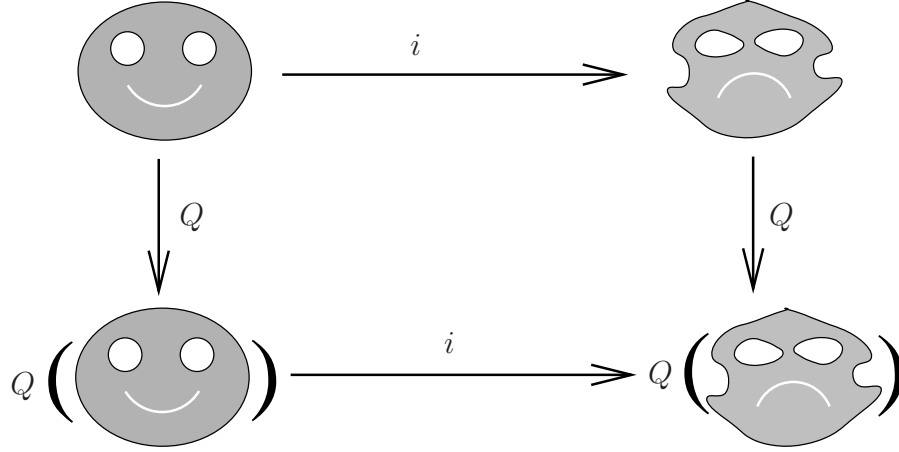


Figure 1.15. The query  $Q$  is topological.

We remark that deciding whether a query is topological is undecidable. For completeness we give the proof first presented in ?.

**THEOREM 1.54** *Testing whether a query expressible in  $\text{FO}(+, \times, <, 0, 1)$  is topological is undecidable.*

*Proof* Let  $\mathcal{S}$  be the database schema consisting of a single unary relation  $S$ . For other schemas, the proof is similar. We will reduce the problem of deciding the truth of sentences of the  $\forall^*$ -fragment of number theory to the problem of deciding whether a query is topological. The  $\forall^*$ -fragment of number theory is known to be undecidable since Hilbert's 10th problem can be formulated in it.

We encode a natural number  $n$  by the unary finite relation  $\text{enc}(n) = \{0, 1, 2, \dots, n\}$ . A  $k$ -dimensional vector of natural numbers  $(n_1, n_2, \dots, n_k)$  is encoded by the relation

$$\begin{aligned} \text{enc}(n_1, n_2, \dots, n_k) = & \text{enc}(n_1) \cup (\text{enc}(n_2) + n_1 + 2) \cup \\ & \dots \cup (\text{enc}(n_k) + n_1 + 2 + \dots + n_{k-1} + 2). \end{aligned}$$

For a fixed  $k$ , the corresponding decoding is expressible in  $\text{FO}(+, \times, <, 0, 1)$  (the reader might want to verify this).

We now associate with each first-order sentence

$$\forall x_1 \forall x_2 \dots \forall x_n \varphi(x_1, \dots, x_n)$$

of number theory the following query  $Q_\varphi$  expressed by the  $\text{FO}(+, \times, <, 0, 1)$  formula over  $\mathcal{S}$ :

$$Q_\varphi = \text{if } S \text{ encodes a vector } (n_1, \dots, n_k) \in \mathbb{N}^k \text{ then} \\ \text{if } \varphi(n_1, \dots, n_k) \text{ then return } \emptyset \\ \text{else return } 0 \\ \text{else return } \emptyset.$$

It is easily verified that  $Q_\varphi$  is topological if and only if the sentence  $\forall x_1 \forall x_2 \cdots \forall x_n \varphi(x_1, \dots, x_n)$  is true. Therefore, if testing whether a query expressible in  $\text{FO}(+, \times, <, 0, 1)$  is topological would be decidable, so would be the  $\forall^*$ -fragment of number theory. QED

**EXERCISE 1.55** Prove the claim in the above proof that  $Q_\varphi$  is topological if and only if the sentence  $\forall x_1 \forall x_2 \cdots \forall x_n \varphi(x_1, \dots, x_n)$  is true.

In the remainder of this section we investigate which databases can be distinguished by means of topological queries expressible in  $\text{FO}(+, \times, <, 0, 1, S)$ .  $\text{FO}(+, \times, <, 0, 1, S)$ . By definition, isotopic databases cannot be distinguished in such a way. It turns out that reverse direction does not hold. In general, it is not known when two databases are distinguishable by topological queries. There are two exceptions however. The first exception is for databases consisting of a single closed semi-algebraic set, the second exception is for databases consisting of possibly many semi-algebraic sets, but in which only points of “regular” cone types are allowed. The latter case is discussed in (?). We will only describe the case of closed databases. The following results are taken from (?).

**Cut- and glue transformations and the first-order inexpressibility of connectivity.** Here we describe two transformations on closed semi-algebraic sets in  $\mathbb{R}^2$ . We call these transformations the *cut* and the *glue transformation*. To apply the cut transformation to a set  $A \subset \mathbb{R}^2$ , one first needs to create locally (via a rubber-sheet transformation of the plane) a rectangular strip in  $A$  and then perform a cut as illustrated by the left to right direction in Figure ???. So, this cut removes a rectangular part of the strip and perforates one of the remaining ends. The glue transformation is the inverse of the cut (illustrated by the right to left arrow in Figure ??).

We will show that whenever two planar sets differ from each other by a *cut*- or *glue transformation*, they cannot be distinguished by a topological query expressed by a sentence in  $\text{FO}(+, \times, <, 0, 1, S)$ .



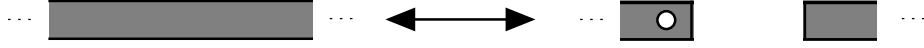


Figure 1.16. The cut and glue transformations on figures in  $\mathbb{R}^2$ .

**THEOREM 1.56** *Let  $A$  and  $B$  be closed semi-algebraic sets in  $\mathbb{R}^2$ . If  $B$  is obtained from  $A$  by a cut- or glue transformation, then  $A$  and  $B$  are indistinguishable by a  $\text{FO}(+, \times, <, 0, 1, S)$ -sentence that expresses a topological query.*

*Proof* Assume, for the sake of contradiction, that there exist closed semi-algebraic sets  $A$  and  $B$  that differ by one cut- or glue transformation but which can be distinguished by a first-order expressible topological sentence. Hence, there exists a first-order sentence  $\varphi$ , which expresses a topological query, such that  $\varphi(A) = \text{TRUE}$  and  $\varphi(B) = \text{FALSE}$ .

Consider the decision problem MAJORITY about two finite sets of reals  $S_1$  and  $S_2$  (see above). We will prove the existence of a formula  $\psi(x, y)$  in  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$  such that for any finite database  $D = (D_1, D_2)$  over  $(S_1, S_2)$ , we have that for  $E_A(D) = \{(x, y) \in \mathbb{R}^2 \mid (\mathbb{R}, D) \models \psi(x, y)\}$ ,  $\varphi(E_A(D)) = \text{TRUE}$  if and only if MAJORITY( $D_1, D_2$ ) is FALSE.

By Lemma ??, this then yields the desired contradiction. This reduction technique is inspired by (?).

Obviously, for any finite  $D = (D_1, D_2)$ , the part  $D_1 \subseteq D_2$  can be tested in first-order logic. For given  $D_1 = \{r_1, \dots, r_n\}$  and  $D_2 = \{a_1, \dots, a_m\}$  with  $0 < r_1 < \dots < r_n$  and  $0 < a_1 < \dots < a_m$ , we construct within the fixed rectangular part  $\alpha$  of  $\mathbb{R}^2$ , where the cut-or-glue transformation takes place, a closed semi-algebraic set  $E(D)$  consisting of interconnected strips.

This construction is similar to constructions in (?) and is illustrated in Figure ?? for  $n = 6$  and  $m = 4$ . The construction is as follows. Take a rectangular subarea  $\alpha'$  of  $\alpha$ . Let  $(b_0, s_0)$  be the left bottom corner of  $\alpha'$  and let  $h$  and  $w$  be its height and width. Then sets  $D'_1 = \{s_0, \dots, s_n\}$  and  $D'_2 = \{b_0, b_1, \dots, b_m, b_{m+1}, \dots, b_{2m}\}$ , with  $s_i = s_0 + r_i h / r_n$  ( $0 < i \leq n$ ),  $b_i = b_0 + a_i w / 2a_m$  and  $b_{m+i} = b_i + w/2$  ( $0 < i \leq m$ ) are constructed. Then, the following closed strips of  $E(D)$  are constructed:

1. the filled convex quadrangle with corners  $(b_i, s_j)$ ,  $((b_i + b_{i+1})/2, s_j)$ ,  $(b_{i+1}, s_{j+1})$ ,  $((b_{i+1} + b_{i+2})/2, s_{j+1})$  for  $0 < i < 2m - 1$  and  $0 \leq j < n$  and for  $i = j = 0$ ,

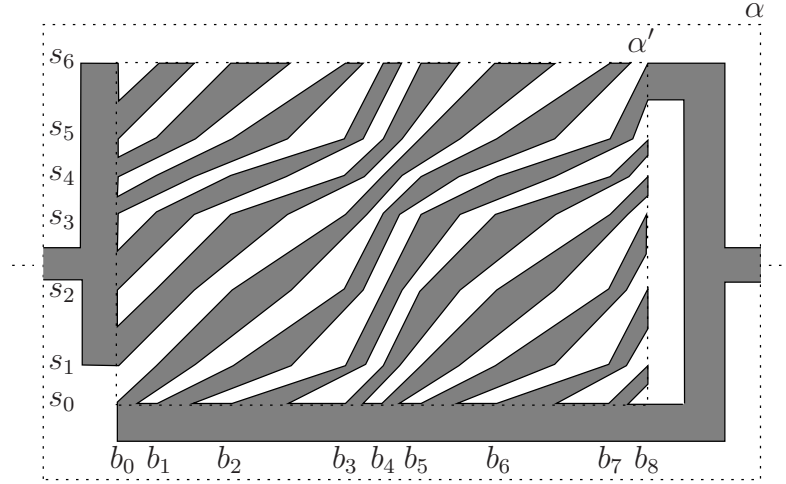


Figure 1.17. Construction of  $E(D)$  for  $D = (D_1, D_2)$  with  $D_1 = \{1, 3, 5, 6, 7, 9\}$  and  $D_2 = \{1, 3, 6, 7\}$  in the rectangular area  $\alpha$ .

2. the filled convex quadrangle with corners  $(b_{2m-1}, s_j)$ ,  $((b_{2m-1} + b_{2m})/2, s_j)$ ,  $(b_{2m}, s_{j+1})$ ,  $(b_{2m}, (s_j + s_{j+1})/2)$  for  $0 \leq j < n$ ,
3. the filled convex quadrangle with corners  $(b_0, (s_{j+1} + s_{j+2})/2)$ ,  $((b_1 + b_2)/2, s_{j+2})$ ,  $(b_1, s_{j+2})$ ,  $(b_0, s_{j+1})$  for  $0 \leq j < n - 1$ .

Finally, a number of additional closed strips are added in the area  $\alpha \setminus \alpha'$  (as illustrated in Figure ??) to complete the construction of  $E(D)$ . Remark that the complete construction of  $E(D)$ , as described above, starting from  $D_1$  and  $D_2$  can be expressed for any  $D = (D_1, D_2)$  by a formula  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$ .

We then glue  $E(D)$  to the part of  $A$  outside the cut-or-glue transformation area  $\alpha$  and denote the resulting set by  $E_A(D)$ . Note that outside this area,  $E_A(D)$  and  $B$  are identical.

Hence, there exists a formula  $\psi$  in  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$  such that for any  $D$ , we obtain a semi-algebraic set  $E_A(D)$ . Moreover, by construction  $E(D)$  will be homeomorphic to the right part of Figure ?? if  $\text{MAJORITY}(D_1, D_2)$  is true, and homeomorphic to the left part of Figure ?? otherwise.

Hence, in case of majority  $E_A(D)$  is homeomorphic to  $B$ , and in the other case it is homeomorphic to  $A$ . Since  $\varphi$  expresses a topological

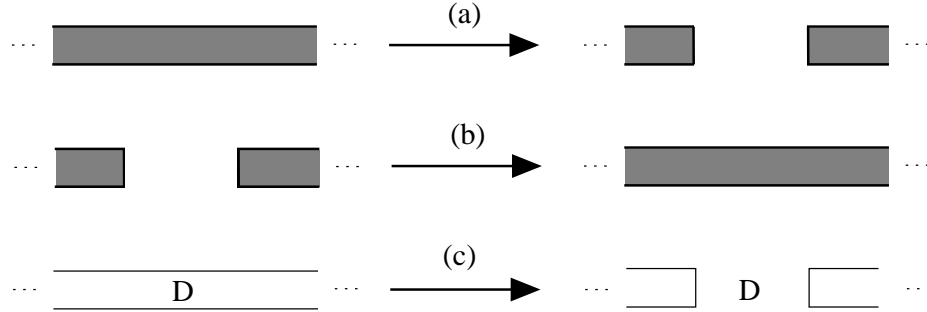


Figure 1.18. Three more cut and glue transformations on figures in  $\mathbb{R}^2$ .

query which distinguishes between  $A$  and  $B$ , we can use  $\varphi$  to express MAJORITY. QED

With some more work additional cut and glue transformations can be proven to produce results that are indistinguishable from the original spatial figure. Three such transformations are shown in Figure ?? . In (a), we have a stip that can be cut (this time without producing a perforation in one of the sides). In (b), we have the reverse transformation of (a) and in (c) we see how parallel lines can be rewired, even when some data  $D$  is in between them.

EXERCISE 1.57 Use this theorem to show that the connectivity query is not expressible in  $\text{FO}(+, \times, <, 0, 1)$ .

EXERCISE 1.58 Let us consider a linear version of the cut- or glue transformation (with a squared hole rather than a circular one). Let  $A$  and  $B$  be closed semi-linear sets in  $\mathbb{R}^2$ . Prove that if  $B$  is obtained from  $A$  by a (linear) cut-or-glue transformation, then  $A$  and  $B$  are indistinguishable by a  $\text{FO}(+, <, 0, 1)$ -sentence that expresses a topological query.

**The point structure of a closed semi-algebraic set in  $\mathbb{R}^2$ .** In the previous section, we have shown that there are non-homeomorphic sets that nevertheless are indistinguishable by  $\text{FO}(+, \times, <, 0, 1)$ -sentences expressing topological properties. More specifically, the cut- or glue transformations relate some indistinguishable databases. However, there are a few other transformations with the same property and the question now rises which databases can be transformed into each other using one of these transformations. For closed databases, we can characterize this exactly.

First, recall the definition ?? of the point structure of a database.

DEFINITION 1.59 Let  $A$  and  $B$  be closed semi-algebraic sets in  $\mathbb{R}^2$ . We say that  $\Pi(A)$  is *isomorphic* to  $\Pi(B)$  (denoted by  $\Pi(A) \cong \Pi(B)$ ) if there is a bijection  $f$  from  $A \cup \{\infty\}$  to  $B \cup \{\infty\}$  with  $f(\infty) = \infty$ , such that  $\Pi(A) = \Pi(B) \circ f$ .

The main result in this context is that indistinguishability by topological queries can be expressed in terms of point-structure isomorphism.

THEOREM 1.60 *Let  $A$  and  $B$  be closed semi-algebraic sets in  $\mathbb{R}^2$ . The sets  $A$  and  $B$  are indistinguishable by topological  $\text{FO}(+, \times, <, 0, 1)$ -queries if and only if  $\Pi(A) \cong \Pi(B)$ .*

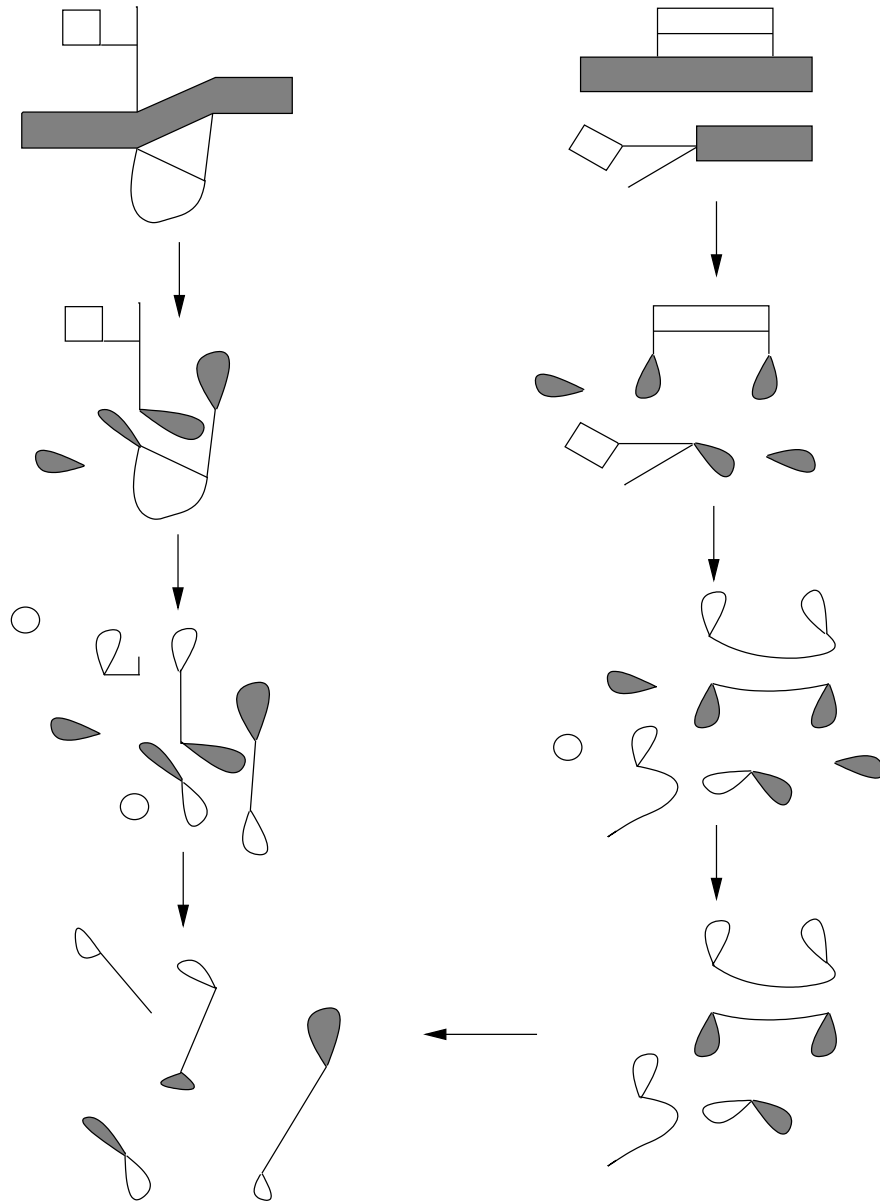
*Proof* We briefly sketch the proof of this theorem. If two closed semi-algebraic sets have a different point structure than they can be distinguished by a topological query that expresses that the one set contains a different number of points than the other set with a specific cone. In Section ??, we will discuss in more detail how a cone of a point can be expressed in  $\text{FO}(+, \times, <, 0, 1)$ .

If two closed semi-algebraic sets in  $\mathbb{R}^2$  have the same point structure, they can be transformed into the same canonical semi-algebraic set using the cut and glue transformations shown in Figure ??. First, two-dimensional lobes are cut around singular points, then the lines are cut into lobes. This produces “flowers” that may be connected by stems. This rewriting process is illustrated in Figure ??. QED

EXERCISE 1.61 Use this theorem to show that the two sets shown in Figure ?? are indistinguishable by a topological query.

One may wonder whether this characterization holds for arbitrary semi-algebraic sets in  $\mathbb{R}^2$  too. This is not the case as it can be shown that the two sets shown in Figure ?? can be distinguished by a topological  $\text{FO}(+, \times, <, 0, 1)$ -sentence even though they have isomorphic cone structures (?).

Theorem ?? gives us an idea of which closed semi-algebraic sets in the plane are distinguishable by topological first-order queries, but it doesn’t give us a full picture of the expressive power of the topological fragment of  $\text{FO}(+, \times, <, 0, 1)$ . There are results that characterize this expressive power, however. It has been shown that the topological fragment of  $\text{FO}(+, \times, <, 0, 1)$  just allows you to formulate queries that talk about types of cones appearing in a semi-algebraic sets and on the number of points having particular cones (?).



*Figure 1.19.* The transformation of two closed semi-algebraic sets (on the left and right hand top) into their canonical form (left bottom).



Figure 1.20. One and two balls cannot be distinguished by a topological  $\text{FO}(+, \times, <, 0, 1)$ -query.

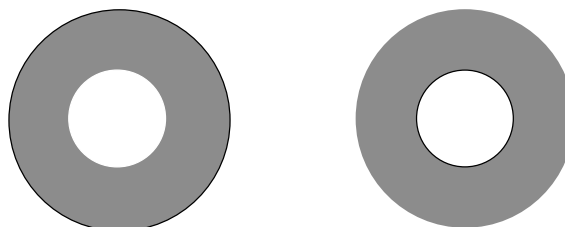


Figure 1.21. Semi-open annuli with the opposite open sides can be distinguished by a topological  $\text{FO}(+, \times, <, 0, 1)$ -query.

## 5.4 Expressing the cone radius

As we have seen, the local conical property of semi-algebraic sets plays a prominent role in the study of first-order expressiveness of topological properties. In this section, we will show that  $\text{FO}(+, \times, <, 0, 1)$  is expressive enough to find a cone radius for each point in the database.

More specifically, the following result holds (?).

**THEOREM 1.62** *There exists an  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$ -formula  $\varphi(r)$ , with  $S_1$  and  $S_2$  of arity  $n$ , that for a semi-algebraic set  $A$  in  $\mathbb{R}^n$  and a point  $\vec{p}$  in  $\mathbb{R}^n$ , defines one  $r$  in  $\mathbb{R}$  for which  $(A, \{\vec{p}\}) \models \varphi(r)$  and which is a cone radius for  $A$  in  $\vec{p}$ .*

To give the complete proof of this theorem would lead us too far. Instead, we provide the intuition behind the proof. Moreover, we only consider the case when  $n = 2$  and assume that  $A$  is a semi-algebraic set consisting only of points of cone type ( $LL$ ).

Consider the semi-algebraic set depicted on the left of Figure ???. On the right of this picture, we have shown the intersections of circles of various radii, centered around  $\vec{p}$  with  $A$ . As can be seen, each time there exists a point  $\vec{q}$  in  $A$  which has a tangent line perpendicular to the line going through  $\vec{p}$  and  $\vec{q}$  (depicted by the dashed circles), the

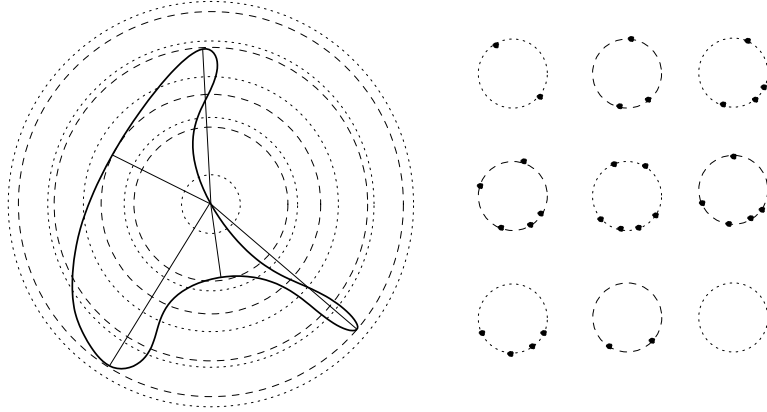


Figure 1.22. Illustration of how the intersections  $S^1(\vec{p}, r) \cap A$  change in terms of  $r$ .

topological type of the intersection, which in this case is nothing else than the number of points, changes.

This observation can be formalized using a variant of the triviality theorem which states that the topological type of  $S^1(\vec{p}, r_1) \cap A$  and  $S^1(\vec{p}, r_2) \cap A$  are the same (meaning that there exists a homeomorphism between these two sets) if for any  $r \in [r_1, r_2]$  there exists no point  $\vec{q}$  such that  $d(\vec{p}, \vec{q}) = r$  and the tangent line in  $\vec{q}$  is perpendicular to  $\vec{q} - \vec{p}$ .

**EXERCISE 1.63** Let  $A \subseteq \mathbb{R}^2$  be a one-dimensional semi-algebraic set. Show that there exists an  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$ -formula  $\varphi$  which returns for any pair  $(A, \vec{p})$  all point  $\vec{q}$  which have a tangent line perpendicular to the vector  $\vec{q} - \vec{p}$ . (*Hint: Use the standard definition of tangent line in terms of the limit of the secant lines.*)

Given the  $\text{FO}(+, \times, <, 0, 1, S_1, S_2)$ -expression  $\varphi$  of the previous exercise, define

$$r_{\vec{p}} = \frac{1}{2} \min\{d(\vec{p}, \vec{q}) \mid (A, \vec{p}) \models \varphi(\vec{q})\}.$$

It is clear that for any  $(A, \vec{p})$ ,  $r_{\vec{p}}$  is expressible by a first-order formula. By definition, there is no point  $\vec{q}$  with a tangent line perpendicular to  $\vec{q} - \vec{p}$  for any  $0 < r \leq r_{\vec{p}}$ . In other words, for all  $0 < r \leq r_{\vec{p}}$ , all intersections  $S^1(\vec{p}, r) \cap A$  are homeomorphic, and it can be shown that  $r_{\vec{p}}$  is indeed a cone radius of  $A$  in  $\vec{p}$ .

We briefly mention that for the general case, we have to decompose  $A$  first into parts on which a tangent space can be defined. Moreover, this has to be shown to be first-order expressible. Next, similar to the case above, a cone radius of a point  $\vec{p}$  can then be defined as a radius

smaller than any point  $\vec{q}$  with a tangent space perpendicular to  $\vec{q} - \vec{p}$ , where the tangent space is taken relative to the decomposition of  $A$ .

### 5.5 The expressiveness of $\text{FO}(+, \times, <, 0, 1)$ and $\text{FO}(+, <, 0, 1)$

We end this section with a remark on the comparison of the expressive powers of  $\text{FO}(+, \times, <, 0, 1)$  and of  $\text{FO}(+, <, 0, 1)$ . We will illustrate that the expressive power of  $\text{FO}(+, <, 0, 1)$  is less than that of  $\text{FO}(+, \times, <, 0, 1)$ .

The topological interior of a two-dimensional set  $S$  can be expressed in  $\text{FO}(+, \times, <, 0, 1)$  by the formula

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

Since the topology of  $\mathbb{R}^2$  based on open discs is equivalent to the one based on open rectangles, we can equivalently express the topological interior of a semi-algebraic subset of  $\mathbb{R}^2$  in  $\text{FO}(+, <, 0, 1)$  by the formula

$$\exists \varepsilon (\varepsilon > 0 \wedge \forall x' \forall y' (|x - x'| < \varepsilon \wedge |y - y'| < \varepsilon \rightarrow S(x', y'))).$$

But there are other queries for which the multiplication seems to be really necessary to express them in first-order logic. If we want to express that a two-dimensional semi-linear set is *convex*, for instance, then we can do this in  $\text{FO}(+, \times, <, 0, 1)$  with the formula

$$\forall \vec{x} \forall \vec{y} (S(\vec{x}) \wedge S(\vec{y}) \rightarrow \forall \lambda (0 \leq \lambda \leq 1 \rightarrow S(\lambda \vec{x} + (1 - \lambda) \vec{y}))).$$

Clearly, in the subexpression  $\lambda \vec{x} + (1 - \lambda) \vec{y}$  multiplication is used and it may seem difficult to imagine that convexity of semi-linear sets might be expressible without multiplication.

But it turns out that we have the following property (?).

**PROPOSITION 1.64** *A semi-linear set of  $\mathbb{R}^n$  is convex if and only if it is closed under taking midpoints.*

**EXERCISE 1.65** Verify this property. Verify also that this property does not hold for subsets of  $\mathbb{R}^n$  that are not semi-linear (think of  $\mathbb{Q}^n$ ).

We can therefore express convexity of semi-linear sets by the  $\text{FO}(+, <, 0, 1)$ -formula

$$\forall \vec{x} \forall \vec{y} (S(\vec{x}) \wedge S(\vec{y}) \rightarrow \exists \vec{z} (2\vec{z} = \vec{x} + \vec{y} \wedge S(\vec{z}))).$$

Now we can wonder whether all  $\text{FO}(+, <, 0, 1)$  expressible queries and properties are also expressible in  $\text{FO}(+, \times, <, 0, 1)$ . Clearly,  $\text{FO}(+, \times, <, 0, 1)$



is more expressive than  $\text{FO}(+, <, 0, 1)$  for what concerns queries that return some  $n$ -dimensional result. The constant query that returns on any input the  $n$ -dimensional unit sphere, for instance, is not expressible in  $\text{FO}(+, <, 0, 1)$ . But what about properties? Are  $\text{FO}(+, \times, <, 0, 1)$  and  $\text{FO}(+, <, 0, 1)$  equally expressive for what concerns properties?

The answer is negative, however, as is illustrated by the following theorems. The first theorem is from (?) and was proven using Ehrenfeucht-Fraïssé games.

**THEOREM 1.66** *The boolean query deciding whether a semi-linear set  $S$  contains real numbers  $u$  and  $v$  satisfying  $u^2 + v^2 = 1$  is expressible in  $\text{FO}(+, \times, <, 0, 1, S)$ , but not in  $\text{FO}(+, <, 0, 1, S)$ .*

Another example of a property that is not expressible in  $\text{FO}(+, <, 0, 1)$  is from (?).

**THEOREM 1.67** *The boolean query deciding whether a semi-linear subset  $S$  of  $\mathbb{R}^2$  contains a line is expressible in  $\text{FO}(+, \times, <, 0, 1, S)$ , but not in  $\text{FO}(+, <, 0, 1, S)$ .*

## 6. Extensions of logical query languages

In this section we will extend first-order logic with operators in order to increase its expressive power. We begin with introducing transitive closure logics ( $TC$ ;  $TC$ ) and ( $TC$ ). Next, we extend first-order logic with a while-loop ( $WH$ ). We conclude the section by extending first-order logic with specific operators for topological properties ( $TOP$ ).

### 6.1 First-order logic with all-purpose operators

Let us first make a small digression to the relational database setting. Here, a database  $D$  is a finite model of the signature  $\mathcal{S}$ , where  $\mathcal{S}$  consist of a binary relation name  $S$ . The transitive closure of  $D$  is usually computed in stages  $X_i$ ,  $i = 0, 1, 2, \dots$ . Initially,  $X_0 = D$  and for  $n \geq 0$  we have

$$X_{n+1} = X_n \cup \{(x, y) \mid \exists z (X_n(x, z) \wedge D(z, y))\}.$$

Since the number of pairs in  $D$  is finite and by construction,  $X_n \subseteq X_{n+1}$ , after finitely many steps we end up with the situation that  $X_{n+1} = X_n$ . The fixed-point  $X_n$  is then the transitive closure of  $D$ .

It is well-known that the transitive closure of  $D$  cannot be computed by a query expressible in first-order logic over  $\mathcal{S}$ . In order to be able to express the transitive closure of  $D$  one has studied the extension of first-order logic with transitive closure ( $TC$ ).

EXAMPLE 1.68 We show that transitive closure logics can express that a graph  $G = (V, E)$  is connected. Indeed, let  $D$  be the (binary) database containing the edge relation  $E$  of  $G$ . If we interpret the formula

$$\text{TC}_{x;y}S(x, y)$$

as an expression which evaluates on  $D$  to the transitive closure of  $D$ , then the expression

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

evaluates to true on  $D$  if and only if the corresponding graph  $G$  is a connected, i.e., if for any two  $s, t$  in the domain of  $D$ , i.e., the set of vertices  $V$  of  $G$ , there exists a sequence  $(v_0, v_1), (v_1, v_2), \dots, (v_{n-1}, v_n)$  such that  $D(v_i, v_{i+1})$  for  $i = 0, 1, \dots, n-1$  and  $s = v_0$  and  $t = v_n$ .

Motivated by this relational example, we start by adding the transitive closure operator to first-order logic in the constraint database setting. The resulting query language will be denoted by  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$ .

**First-order logic with transitive-closure operator.** A formula in  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  is a formula built in the same way as an  $\text{FO}(+, \times, <, 0, 1)$  formula, but with the following extra formation rule: if  $\psi(\vec{x}, \vec{y})$  is a formula with  $\vec{x}$  and  $\vec{y}$   $k$ -tuples of real variables, with all free variables of  $\psi$  among  $\vec{x}, \vec{y}$ , and if  $\vec{s}, \vec{t}$  are  $k$ -tuples of real variables, then

$$[\text{TC}_{\vec{x};\vec{y}}\psi(\vec{x}, \vec{y})](\vec{s}, \vec{t}) \quad (1.1)$$

is also a formula which has as free variables those in  $\vec{s}$  and  $\vec{t}$ .

We will distinguish between  $\text{FO}(+, <, 0, 1) + \text{TC}$  and  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  depending on whether only linear (i.e.,  $\times$  is not allowed) or also non-linear constraints are allowed.

To explain the semantics of a subformula of the above form (??), we compute again the following stages

$$X_0 = \psi(D), \quad X_{n+1} = X_n \cup \{(\vec{x}, \vec{y}) \mid \exists \vec{u} (X_n(\vec{x}, \vec{u}) \wedge X_0(\vec{u}, \vec{y}))\},$$

until the fixed-point, which we denote by  $X_\infty$ , is reached. Then the semantics of  $[\text{TC}_{\vec{x};\vec{y}}\psi(\vec{x}, \vec{y})](\vec{s}, \vec{t})$  is defined as  $(\vec{s}, \vec{t})$  belonging to the  $2k$ -ary relation  $X_\infty$ .

The question is now how a formula in  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  is evaluated. Assume that  $\psi$  is an  $\text{FO}(+, \times, <, 0, 1)$ -formula. Then we simply can compute the quantifier-free description of  $X_{n+1}$  recursively by evaluating the corresponding  $\text{FO}(+, \times, <, 0, 1)$ -expressions. After each

computation we then test whether  $X_n = X_{n+1}$ . If this holds, we have obtained the fixed-point and test whether  $(\vec{s}, t)$  is in  $X_n$ .

In general, the semantics of a formula  $\varphi$  in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  is evaluated in the standard bottom-up fashion. The result of the evaluation of subformulas is passed on to formulas that are higher up in the parsing tree of  $\varphi$ .

However, in this context, 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 computation of  $X_\infty$  might not terminate. Hence, the property of effective computability of queries, expressible in  $\text{FO}(+, \times, <, 0, 1)$  or  $\text{FO}(+, <, 0, 1)$ , is lost when extending these logics with recursion. In case the computation of  $X_\infty$  does not terminate, then the semantics of the formula (??) (and any other formula in which it occurs as subformula) is undefined.

We therefore have to distinguish between formulas for which the evaluation terminates, we call such formulas *terminating*, and formulas for which the evaluation does not terminate, i.e., the *non-terminating* formulas. To illustrate this difference, we now provide an example of a terminating and non-terminating formula in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$ .

EXAMPLE 1.69 Let  $\mathcal{S}$  consist of the binary relation  $S$ . Consider the  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  formula

$$[TC_{x;y}S(x, y)](s, t)$$

and let  $D = \{(x, y) \mid y = 2x\}$ . We have for each  $n \geq 0$ ,

$$X_n = \{(x, y) \mid \exists i \in \mathbb{N}, i \leq n, y = 2^i x\}.$$

It is clear that for all  $n \geq 0$ ,  $X_n \neq X_{n+1}$  and hence we need to compute infinitely many stages until the fixed point  $X_\infty$  is reached. We have depicted this example in ??.

EXERCISE 1.70 Show that  $X_\infty$  in the previous example is not a semi-algebraic set.

On the other hand, even when  $X_\infty$  is semi-algebraic, its computation may still be non-terminating.

EXERCISE 1.71 Give an example of a non-terminating transitive closure for which  $X_\infty$  is a semi-algebraic set. (*Hint*: Look at Example ?? for inspiration.)

One may wonder whether the non-termination of the  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  formula in Example ?? is caused by the unboundedness of the input database.

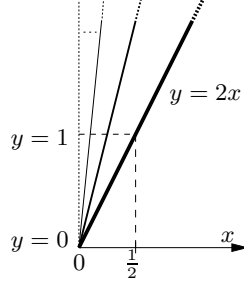


Figure 1.23. Computation of  $[TC_{x;y}S(x,y)](s,t)$  for  $D = \{(x,y) \mid y = 2x\}$  (thickest line). Consecutive stages are drawn in decreasingly finer lines.

EXERCISE 1.72 Provide an example of a bounded database  $D$  on which the  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  expression  $[TC_{x;y}S(x,y)](s,t)$  does not terminate.

Since  $\text{FO}(+, \times, <, 0, 1)$  is a sub-language of  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$ , there are infinitely many terminating formulas. We now give an example of a terminating formula which is not in  $\text{FO}(+, \times, <, 0, 1)$ .

EXAMPLE 1.73 Let the database schema  $\mathcal{S}$  consist of the unary relation name  $S$ . Consider the formula

$$[TC_{x;y}\varphi(r, x, y) \wedge S(r)](s, t)$$

where  $\varphi(r, x, y)$  defines the graph of the continuous piecewise affine function that maps  $x$  to

$$y = \begin{cases} 0 & \text{if } 0 \leq x \leq \frac{1}{r}, \\ x - \frac{1}{r} & \text{if } \frac{1}{r} < x < 1, \\ 1 - \frac{1}{r} & \text{if } x = 1. \end{cases}$$

Then, for  $D = \{p\}$  and  $p \in \mathbb{N}$  we have for each  $n \geq 0$ ,

$$X_n = \bigcup_{i=1}^n \{(x, y) \mid \varphi(\frac{n}{p}, x, y)\}.$$

It is clear that  $X_{p+1} = X_p$  and hence this  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  formula is terminating. Moreover, it expresses the query  $Q_{\text{nat}}$  which tests whether the number stored in a database is a natural number. We have illustrated this example in Figure ??.

EXERCISE 1.74 Show that the query  $Q_{\text{nat}}$  cannot be expressed in the logic  $\text{FO}(+, \times, <, 0, 1)$ .

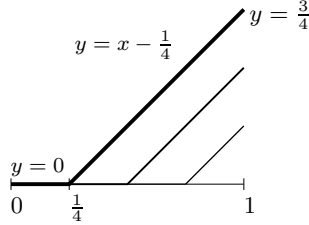


Figure 1.24. Example of the terminating  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  formula of Example ?? for  $D = \{4\}$ .

As a result of the above exercise and Example ??, we have that  $\text{FO}(+, \times, <, 0, 1)$  is strictly less expressive than  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$ .

**EXERCISE 1.75** Give an example of an unbounded database on which the  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  expression  $[TC_{x;y}S(x, y)](s, t)$  terminates. (*Hint:* Use the formula  $\varphi(r, x, y)$  of the previous example.)

As explained in Example ??, transitive closure logic can express the connectivity of finite graphs. Similarly,  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  can express the connectivity of constraint databases. We first consider the case of linear constraints.

Let  $\mathcal{S}$  be a schema with one relation name  $S$  of arity  $n$ . Consider the following  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  formula  $\text{connected}(S)$ :

$$\forall \vec{s} \forall \vec{t} ((S(\vec{s}) \wedge S(\vec{t})) \rightarrow [\text{TC}_{\vec{x};\vec{y}} \text{lineconn}](\vec{s}, \vec{t}))$$

where  $\text{lineconn}(\vec{x}, \vec{y})$  is the formula

$$\forall \lambda (0 \leq \lambda \leq 1 \wedge \forall \vec{t} (\vec{t} = \lambda \vec{x} + (1 - \lambda) \vec{y} \rightarrow S(\vec{t})).$$

We now claim that a pair of points  $(\vec{p}, \vec{q})$  belongs to transitive closure of  $\text{lineconn}(D)$  (with  $D$  semi-linear) if and only if  $\vec{p}$  and  $\vec{q}$  belong to the same connected component. Hence, if we can show that  $\text{connected}$  is a terminating formula, then this implies that  $\text{connected}$  expresses connectivity of semi-linear sets.

**THEOREM 1.76** *The formula  $\text{connected}$  terminates on all linear constraint databases  $D$  over  $\mathcal{S}$  and correctly expresses connectivity of semi-linear sets.*

*Proof* Since  $D$  is semi-linear, two points  $\vec{p}$  and  $\vec{q}$  belong to the same connected component of  $D$  if and only if there exists a piecewise linear path from  $\vec{p}$  to  $\vec{q}$  lying entirely in  $D$ . This follows directly from the semi-linear version of Theorem ?? and Exercise ?. So, indeed

$[\text{TC}_{\vec{x};\vec{y}} \text{lineconn}](\vec{s}, \vec{t})$  holds if and only if  $\vec{s}$  and  $\vec{t}$  belong to the same connected component of  $D$ .

To conclude that the evaluation of the transitive closure in the formula **connected** ends in finitely many steps, we need to show that there exists an upper bound on the number of line segments in  $S$ , needed to connect any two points in the same connected component of  $S$ . Now, any semi-linear set can be decomposed in a finite number of convex sets (?, Chapter 8, Exercise 2.14 (2)). The finiteness of this decomposition yields the desired bound since any two points in a convex set are connected by a single straight line segment. We have illustrated this in Figure ?? for  $n = 2$ . QED

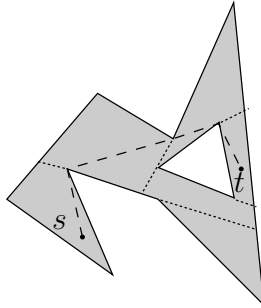


Figure 1.25. A semi-linear set decomposed in convex sets. The boundaries of the convex sets are shown in dotted lines. We have depicted a piecewise linear path (dashed line) between points  $s$  and  $t$ .

The above  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$ -formula **connected** cannot be applied to arbitrary semi-algebraic inputs, still guaranteeing termination. Indeed, when we apply it to the binary relation, depicted in Figure ?? consisting of the points lying strictly between the parabola  $y = x^2$  and the translated one  $y = x^2 + 1/2$ .

Although any two points in this set can be connected by a finite number of line segments, there is no upper bound on the number of segments needed to connect two points.

There are also examples of semi-algebraic sets  $s$  for which not any two points in a connected component can be connected by a finite piecewise linear path. Here we take as example the set defined by  $(y^3 - x^2 \geq 0 \wedge x < 0 \wedge y < 1) \vee (x = 0 \wedge y = 0)$ , depicted in Figure ?. The cusp point  $(0,0)$  at the bottom cannot be connected by a line segment to any point in the interior of this set. So, the query expressed by **connected** terminates after two iterations, but it does not contain all pairs of points that are in the connected component. This is



Figure 1.26. A semi-algebraic set, the points lying strictly between the parabola  $y = x^2$  and the translated parabola  $y = x^2 + 1/2$ , on which termination is not satisfied.

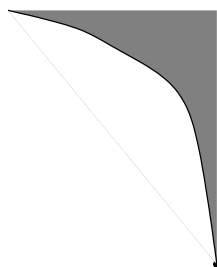


Figure 1.27. A semi-algebraic set with a cusp point.

obviously because piecewise-linear connectivity does not correspond to connectivity for arbitrary semi-algebraic sets.

Basically, Figures ?? and ?? illustrate the only two cases where the **connected** query does not work correctly (i.e., cusp points and cusp points towards  $\infty$ ).

Nevertheless, we can also express connectivity of arbitrary semi-algebraic sets. The proof of this result is a reduction to the linear case. This reduction is possible by the following result.

**THEOREM 1.77** *There exists a terminating  $\text{FO}(+, \times, <, 0, 1) + \text{TC}$  formula which expresses the linearization query “Return a semi-linear set which is homeomorphic to the database.”*

We remark that by the triangulation theorem (Theorem ??), the existence of such semi-linear set is guaranteed.

So, given a database  $D$ , we first apply the linearization query and then apply the **connected** query. This clearly results in the desired  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  formula expressing connectivity.

**EXERCISE 1.78**

- 1 Give an example of a non-linear semi-algebraic set  $D$  on which the  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  formula **connected** does not terminate.
- 2 Give an example of a finite database  $D$  on which the  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  formula **connected** does not terminate.

So far, we have shown individual queries which can be expressed in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$ . Moreover, we have seen that not all formulas in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  are terminating. We now describe two ways of controlling this termination and its effect on the expressiveness.

**Transitive Closure with stop conditions.** A formula in  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  is built in the same way as an  $\text{FO}(+, \times, <, 0, 1)$  formula, but with the following extra formation rule: if  $\psi(\vec{x}, \vec{y})$  is a formula with  $\vec{x}$  and  $\vec{y}$   $k$ -tuples of real variables;  $\sigma$  is an  $\text{FO}(+, \times, <, 0, 1)$  sentence over the input database and a special  $2k$ -ary relation name  $X$ ; and  $\vec{s}, \vec{t}$  are  $k$ -tuples of real variables, then

$$[\text{TC}_{\vec{x};\vec{y}} \psi(\vec{x}, \vec{y}) \mid \sigma](\vec{s}, \vec{t}) \quad (1.2)$$

is also a formula which has as free variables those in  $\vec{s}$  and  $\vec{t}$ . We call  $\sigma$  the *stop condition* of this formula.

The semantics of a subformula of the above form (??) evaluated on databases  $D$  is defined in the same manner as in the case without stop condition, but now we stop not only in case an  $i$  is found such that  $X_i = X_{i+1}$ , but also when an  $i$  is found such that  $(D, X_{i+1}) \models \sigma$ , whichever case occurs first. As above, we also consider the restriction  $\text{FO}(+, <, 0, 1)+\text{TCS}$ .

**EXAMPLE 1.79** As an example of an  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  formula over a two-dimensional input database  $S$ , we take

$$[\text{TC}_{x;y} S(x, y) \mid \exists x \exists y (X(x, y) \wedge y = 1 \wedge 10x \leq 1)](s, t).$$

Here the stop condition is  $\sigma \equiv \exists x \exists y (X(x, y) \wedge y = 1 \wedge 10x \leq 1)$ . When applied to the graph of the function shown in Figure ??, we see that  $X_3$  satisfies the sentence in the stop condition since for instance  $(\frac{1}{16}, 1)$  belongs to it. The evaluation has become terminating (as opposed to the expression without stop condition in Example ??). On input the graph of the function shown in Figure ??, this expression still terminates



after four iterations (since  $X_5 = X_4$ , not because the stop condition is satisfied) and returns the same result as in the case without stop condition.

EXERCISE 1.80 Give examples of stop conditions to ensure termination in Example ??.

**Transitive Closure with start points and parameters.** We can also allow parameters in the transitive closure and restrict the computation of the transitive closure to certain paths, after specifying starting points. We denote the resulting logic with  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$ . A formula in  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  is built exactly as in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  with the exception that parameters  $\vec{u}$  are allowed, i.e.,

$$[\text{TC}_{\vec{x};\vec{y}}\psi(\vec{x}, \vec{y}, \vec{u})](\vec{s}, \vec{t}) \quad (1.3)$$

has as free variables  $\vec{u}$ ,  $\vec{s}$  and  $\vec{t}$ .

The semantics of a subformula of the form (??), with  $\vec{s} = (s_1, \dots, s_k)$ , evaluated on a database  $D$  is defined in the following operational manner: Let  $I$  be the set of indices  $i$  for which  $s_i$  is a constant. Then we start computing the following iterative sequence of  $(2k + \ell)$ -ary relations:

$$X_0 := \psi(D) \wedge \bigwedge_{i \in I} (s_i = x_i)$$

and

$$X_{i+1} := X_i \cup \{(\vec{x}, \vec{y}, \vec{u}) \in \mathbb{R}^{2k+\ell} \mid \exists \vec{z} (X_i(\vec{x}, \vec{z}, \vec{u}) \wedge \psi(\vec{z}, \vec{y}, \vec{u}))\}$$

and stop as soon as  $X_i = X_{i+1}$ . The semantics of

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

is then defined as  $(\vec{s}, \vec{t}, \vec{u})$  belonging to the  $(2k + \ell)$ -ary relation  $X_i$ .

EXAMPLE 1.81 As an example of an  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  formula over a two-dimensional input database  $D$ , we take

$$[\text{TC}_{x;y}S(x, y)](\frac{1}{4}, t).$$

When applied to the graph of the function, shown in Figure ??, we see that  $X_0 = D \cap \{(x, y) \mid x = \frac{1}{4}\}$  and this set is just  $\{(\frac{1}{4}, \frac{1}{2})\}$ . Next,  $X_1$  is computed to be  $\{(\frac{1}{4}, \frac{1}{2})\} \cup \{(\frac{1}{4}, 1)\}$ . In subsequent iterations, no further tuples are added (i.e.,  $X_2 = X_1$ ). This example shows that in

$\text{FO}(+, \times, <, 0, 1)+\text{KTC}$ , the evaluation can be restricted to the computation of certain paths in the transitive closure and this gives control over the termination.

**EXERCISE 1.82** Give an example of an  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  variant of the  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  query of Example ?? with terminating evaluation.

We will also consider the fragment  $\text{FO}(+, <, 0, 1)+\text{KTC}$  of this language.

## 6.2 Computational Completeness

One may wonder what the expressive power of the transitive-closure logics is. It turns out that adding stop conditions or allowing start points and parameters drastically increases the expressive power. More specifically, both  $\text{FO}(+, <, 0, 1)+\text{TCS}$  and  $\text{FO}(+, <, 0, 1)+\text{KTC}$  are computationally complete on semi-linear constraint databases.

This means that for every partial computable (in the sense of Turing computable) query  $Q$  on semi-linear databases, there exists a formula  $\varphi$  in  $\text{FO}(+, <, 0, 1)+\text{TCS}$  (respectively in  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$ ) such that for each semi-linear database  $D$ ,  $\varphi(D)$  is defined if and only if  $Q(D)$  is defined, and in this case  $\varphi(D)$  and  $Q(D)$  are equal.

**THEOREM 1.83** *Both  $\text{FO}(+, <, 0, 1)+\text{TCS}$  and  $\text{FO}(+, <, 0, 1)+\text{KTC}$  are computationally complete on linear constraint databases.*

We remark that this holds only for linear databases. However, a similar result holds on arbitrary databases for the extension  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  of  $\text{FO}(+, \times, <, 0, 1)$  with a while-loop.

**An extension of  $\text{FO}(+, \times, <, 0, 1)$  with a while-loop.** A program in  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  is a finite sequence of *assignment statements* and *while-loops*. Each assignment statement has the form

$$R := \{(x_1, \dots, x_k) \mid \varphi(x_1, \dots, x_k)\},$$

where  $\varphi$  is an  $\text{FO}(+, \times, <, 0, 1)$  formula that uses the relation names  $S_i$  (of the input database schema  $\mathcal{S} = \{S_1, \dots, S_n\}$ ) and previously introduced relation names. Each while-loop has the form

$$\text{WHILE } \varphi \text{ DO } P \text{ OD},$$

where  $P$  is a  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  program and  $\varphi$  an  $\text{FO}(+, \times, <, 0, 1)$  formula that uses relation names  $S_i$  from the input schema and previously introduced relation names.

The semantics of a program applied to a spatial databases is the operational, step by step execution. So, the effect of an assignment statement is to evaluate the  $\text{FO}(+, \times, <, 0, 1)$  formula on the right-hand side on the constraint database  $D$  augmented with the previously assigned-to relation variables, and to assign the result of the evaluation to the relation variable on the left-hand side. The effect of a while-loop is to execute the body as long as non-halting condition  $\varphi$  evaluates to true. One relation name  $R_{\text{out}}$  is designated as the output relation and when the  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  program terminates, the current value of  $R_{\text{out}}$  is considered to be the output.

Again, we have the problem of non-terminating while loops, as the following example shows.

EXAMPLE 1.84 Let  $D = [0, 1]$  and consider the following  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  program  $P$  over the input schema  $\mathcal{S} = \{S\}$ , with  $S$  unary:

```

R := {(x) | S(x)};
Y := ∅;
WHILE R ≠ ∅ DO
  Y := {(x) | φ(x, R)};
  R := R \ Y;
OD

```

where  $\varphi$  is an  $\text{FO}(+, \times, <, 0, 1)$  such that when it is evaluated on any finite sequence of closed non-overlapping intervals  $\{[a_i, b_i]\}$  it results in the sequence of closed non-overlapping intervals  $\{[a_i, a_i + \frac{b_i - a_i}{3}], [a_i + \frac{2(b_i - a_i)}{3}]\}$ . Obviously,  $P$  is non-terminating on  $D$  and  $P(D)$  is undefined. However, if we would allow infinitely long computations, then  $P(D)$  would be the Cantor set which is depicted in Figure ??.



Figure 1.28. The iterative construction of the Cantor set.

EXERCISE 1.85 Consider the following  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  program  $P$  over the input schema  $\mathcal{S} = \{S\}$ , with  $S$  binary:

```

R1 := {(x, y) | S(x, y)};
R2 := {(x, y) | (∃z)(R1(x, z) ∧ S(z, y))};
WHILE R1 ≠ R2
  DO
    R1 := {(x, y) | S(x, y)};
    R2 := {(x, y) | ∃z(R1(x, z) ∧ S(z, y))};
  OD

```

Which query does this program implement? Give an input on which  $P$  terminates and one on which it does not. If  $S$  in the last assignment would be replaced by  $R_1$ , what would the resulting program do?

The previous example shows that in contrast to the transitive-closure logics, an  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  program can recursively reduce the size of relations. This already hints that  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  is more powerful. Indeed, we have the following result.

**THEOREM 1.86** *The languages  $\text{FO}(+, \times, <, 0, 1)+\text{WHILE}$  is computationally complete on constraint databases.*

The only result known for transitive-closure logics on arbitrary databases follows from the completeness on linear databases (Theorem ??) and the fact the linearization query is expressible in  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  (Theorem ??):

**THEOREM 1.87** *The languages  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  and  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  are computationally complete on constraint databases as far as Boolean topological queries is concerned.*

Indeed, given a partial computable Boolean topological query  $Q$  on a database  $D$ , we can apply the linearization query to get  $\widehat{D}$ , that contains linearizations of all relations in  $D$ . By definition of a topological query,  $Q(D)$  and  $Q(\widehat{D})$  are equal. Since there exists an  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  (respectively  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$ ) formula  $\varphi$  expressing  $Q$  on linear database, we can express  $Q$  on  $D$  in  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  (respectively  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$ ) by  $\varphi(\widehat{D})$ .

There are many open problems related to these recursive extensions of  $\text{FO}(+, \times, <, 0, 1)$ . For example, it is not known whether  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  is less expressive than  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  or  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  and it is also unknown whether  $\text{FO}(+, \times, <, 0, 1)+\text{TCS}$  and  $\text{FO}(+, \times, <, 0, 1)+\text{KTC}$  are complete on arbitrary databases.

So far, we have been extending  $\text{FO}(+, \times, <, 0, 1)$  with generic operators expressing many (in some case every) queries inexpressible in  $\text{FO}(+, \times, <, 0, 1)$ . As a result, queries could be undefined because of non-terminating computations.

In the next section we will show how  $\text{FO}(+, \times, <, 0, 1)$  can be extended with specific operators aimed to express specific queries and at the same time guarantee closure.

### 6.3 Extensions of $\text{FO}(+, \times, <, 0, 1)$ with topological operators

We have seen that connectivity of a databases is a desired property which one would like to see being expressible in a query language. We have seen that  $\text{FO}(+, \times, <, 0, 1)$  lacks the power to express this query, while  $\text{FO}(+, \times, <, 0, 1)+\text{TC}$  provides enough power to do so.

However, it is possible to extend  $\text{FO}(+, \times, <, 0, 1)$  directly with a connectivity operator. More generally, define a topological property  $\text{Top}$  as a collection  $\{\mathcal{T}_1, \dots, \mathcal{T}_n, \dots\}$  where  $\mathcal{T}_n$  is a family of sets in  $\mathbb{R}^n$  such that if  $X \in \mathcal{T}_n$ , then for each homeomorphism  $h$  of  $\mathbb{R}^n$ ,  $h(X) \in \mathcal{T}_n$ .

EXAMPLE 1.88 Of special interest is when  $\text{Top}$  expresses the property of being connected. Other examples of  $\text{Top}$  are the property of being closed, being of dimension  $k$ , containing exactly two holes and so on.

Let  $\mathbb{T}$  be a set of topological properties. We then define the language  $\text{FO}(+, \times, <, 0, 1) + \mathbb{T}$  by extending  $\text{FO}(+, \times, <, 0, 1)$  with the following rule: if  $\varphi(\vec{x}, \vec{y})$  is a query then

$$\psi(\vec{x}) \equiv \text{Top}\vec{y} \odot \varphi(\vec{x}, \vec{y})$$

is also a query for any  $\text{Top} \in \mathbb{T}$ . The semantics of such a formula is as follows:  $D \models \varphi(\vec{a})$  iff  $\varphi(\vec{a}, D) = \{\vec{b} \mid D \models \varphi(\vec{a}, \vec{b})\} \in \text{Top}$ .

EXAMPLE 1.89 The query “Is the intersection of regions  $S$  and  $T$  connected” can be written as  $\text{Conn}\vec{x} \odot S(\vec{x}) \wedge T(\vec{x})$ . The query  $\varphi(x) \equiv \text{Conn}(y, z) \odot S(x, y, z)$  returns the set of all  $c \in \mathbb{R}$  for which the intersection of  $S$  with the plane  $x = c$  is a connected set.

THEOREM 1.90 *The logic  $\text{FO}(+, \times, <, 0, 1) + \mathbb{T}$  is closed on constraint databases.*

*Proof* The result follows from a simple induction on the formulas. The only case to prove is  $\psi(\vec{x}) \equiv \text{Top}\vec{y} \odot \varphi(\vec{x}, \vec{y})$  for  $\text{Top} \in \mathbb{T}$ . Let  $\vec{x}$  and  $\vec{y}$  be of length  $n$  and  $m$ , respectively. By induction,  $\varphi(D)$  results in a semi-algebraic set in  $\mathbb{R}^n \times \mathbb{R}^m$ . By the triviality theorem for semi-algebraic sets, there exists a decomposition  $\mathcal{C}$  of  $\mathbb{R}^{n+m}$  into finitely many cells which is trivial over  $\mathbb{R}^n$  and such that  $\varphi(D)$  is the union of cells of  $\mathcal{C}$ . Let  $\mathcal{C}'$  be the projection of  $\mathcal{C}$  onto  $\mathbb{R}^n$ , and let  $C$  be a cell in  $\mathcal{C}'$ . By triviality, for every  $\vec{a}, \vec{b} \in C$ , it is the case that  $(\varphi(D))_{\vec{a}}$  and  $(\varphi(D))_{\vec{b}}$  are homeomorphic, and thus agree on  $\text{Top}$ . Therefore, the output of  $\psi$  on  $D$  is a union of finitely many cells in  $\mathcal{C}'$ . Moreover, since each cell is semi-algebraic, so will be the output  $\psi(D)$ . QED

The linear analog of the previous theorem also holds for the language  $\text{FO}(+, <, 0, 1) + \mathbb{T}$ . However, since the triviality theorem does not hold in this case, one first needs to develop an alternative decomposition theorem (see (?) for details).

## References

- Afrati, F., Cosmadakis, S., Grumbach, S., and Kuper, G. (1994). Linear versus polynomial constraints in database query languages. In Borning, A., editor, *Proceedings of the 2nd Workshop on Principles and Practice of Constraint Programming*, volume 874 of *Lecture Notes in Computer Science*, pages 181–192, Berlin. Springer-Verlag.
- Andradas, C., Bröcker, L., and Ruiz, J. M. (1996). *Constructible sets in real geometry*, volume 33 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag.
- Basu, S., Pollack, R., and Roy, M.-F. (2003a). *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag.
- Basu, S., Pollack, R., and Roy, M.-F. (2003b). *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag.
- Belegardek, O. V., Stolboushkin, A. P., and Taitlin, M. A. (1996). On order-generic queries. Technical Report 96-01, DIMACS.
- Benedetti, R. and Risler, J.-J. (1990). *Real algebraic and semi-algebraic sets*. Actualités Mathématiques. [Current Mathematical Topics]. Hermann.
- Benedikt, M., Dong, G., Libkin, L., and Wong, L. (1996). Relational expressive power of constraint query languages. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems*, pages 5–16.
- Benedikt, M., Grohe, M., Libkin, L., and Segoufin, L. (2003). Reachability and connectivity queries in constraint databases. *J. Comput. System Sci.*, 66(1):169–206.
- Benedikt, M. and Keisler, H. J. (2000). Definability over linear constraints. In Clote, P. and Schwichtenberg, H., editors, *Proceedings of Computer Science Logic, 14th Annual Conference of the EACSL*, volume 1862 of *Lecture Notes in Computer Science*, pages 217–231. Springer-Verlag.
- Benedikt, M., Kuijpers, B., Löding, C, Van den Bussche, J., and Wilke, T. (2006). A characterization of first-order topological properties of planar spatial data. *Journal of the ACM*.

- Benedikt, M. and Libkin, L. (1996). On the structure of queries in constraint query languages. In *11th Annual IEEE Symposium on Logic in Computer Science*, pages 25–34.
- Benedikt, M. and Libkin, L. (2000). Safe constraint queries. *SIAM J. Comput.*, 29(5):1652–1682.
- Benedikt, M.A. and Libkin, L. (1997). Languages for relational databases over interpreted structures. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 87–98.
- Bochnak, J., Coste, M., and Roy, M.-F. (1987). *Géométrie algébrique réelle*. Springer-Verlag.
- Bochnak, J., Coste, M., and Roy, M.-F. (1998). *Real algebraic geometry*, volume 36 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag.
- Caviness, B.F. and Johnson, J.R. (1998). *Quantifier Elimination and Cylindrical Algebraic Decomposition*. New York: Springer-Verlag.
- Codd, E. (1970). A relational model for large shared databanks. *Communications of the ACM*, 13(6):377–387.
- Collins, G.E. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In Brakhage, H., editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, Berlin. Springer-Verlag.
- Coste, M (2000a). *An Introduction to O-minimal Geometry*. Istituti Editoriali e Poligrafici Internazionali, Pisa.
- Coste, M (2000b). *An Introduction to Semialgebraic Geometry*. Istituti Editoriali e Poligrafici Internazionali, Pisa.
- Ebbinghaus, H.-D. and Flum, J. (1995). *Finite model theory*. Perspectives in Mathematical Logic. Springer-Verlag.
- Ebbinghaus, H.-D., Flum, J., and Thomas, W. (1984). *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer-Verlag.
- Geerts, F. (2003). Expressing the box cone radius in the relational calculus with real polynomial constraints. *Discrete Comput. Geom.*, 30(4):607–622.
- Geerts, F. and Kuijpers, B. (2000). Linear approximation of planar spatial databases using transitive-closure logic. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 126–135.
- Geerts, F. and Kuijpers, B. (2005). On the decidability of termination of query evaluation in transitive-closure logics for polynomial constraint databases. *Theoretical Computer Science*, 336(1):125–151. Database Theory—Special issue with selected papers of ICDT’03.
- Giusti, M., Lecerf, G., and Salvy, B. (2001). A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211.

- Grohe, M. and Segoufin, L. (2002). On first-order topological queries. *ACM Transactions on Computational Logic*, 3(3):336–358.
- Grumbach, S. and Su, J. (1995). First-order definability over constraint databases. In *Proceedings of 1st Conference on Principles and Practice of Constraint Programming*, volume 976 of *Lecture Notes in Computer Science*. Springer-Verlag.
- Grumbach, S., Su, J., and Tollu, C. (1995). Linear constraint query languages: expressive power and complexity. In Leivant, D., editor, *Logic and Computational Complexity*, volume 960 of *Lecture Notes in Computer Science*, pages 426–446. Springer-Verlag.
- Gyssens, M., Van den Bussche, J., and Van Gucht, D. (1999). Complete geometric query languages. *J. Comput. System Sci.*, 58(3):483–511.
- Heintz, J., Roy, M.-F., and Solernó, P. (1993). Description of the connected components of a semialgebraic set in single exponential time. *Discrete and Computational Geometry*, 6:1–20.
- Hong, H. (1990). QEPCAD — quantifier elimination by partial cylindrical algebraic decomposition. <http://www.cs.usna.edu/qepcad/B/QEPCAD.html>.
- Kanellakis, P. C., Kuper, G., and Revesz, P. Z. (1995). Constraint query languages. *Journal of Computer and System Sciences*, 51:26–52.
- Kreutzer, S. (2001). Operational semantics for fixed-point logics on constraint databases. In *Logic for programming, artificial intelligence, and reasoning*, volume 2250 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag.
- Kuijpers, B., Paredaens, J., and Van den Bussche, J. (2000). Topological elementary equivalence of closed semi-algebraic sets in the real plane. *J. Symbolic Logic*, 65(4):1530–1555.
- Kuper, G. M., Libkin, L., and Paredaens, J., editors (2000). *Constraint Databases*. Springer-Verlag.
- Motzkin, T. S. (1936). *Beiträge zur Theorie der linearen Ungleichungen*. Doctoral dissertation. Universität Zürich.
- Paredaens, J., Van den Bussche, J., and Van Gucht, D. (1994). Towards a theory of spatial database queries. In *Proceedings of the Thirteenth ACM Symposium on Principles of Database Systems*, pages 279–288.
- Paredaens, J., Van den Bussche, J., and Van Gucht, D. (1995). First-order queries on finite structures over the reals. In *Proceedings of the 10th IEEE Symposium on Logic in Computer Science*, pages 79–89.
- Revesz, R. Z. (2002). *Introduction to Constraint Databases*. Springer-Verlag.
- Rigaux, Ph., Scholl, M., and Voisard, A. (2000). *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann.
- Seidenberg, A. (1954). A new decision method for elementary algebra. *Ann. of Math. (2)*, 60:365–374.



- Stolboushkin, A.P. and Taitslin, M.A. (1996). Linear vs. order constraints over rational databases. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems*, pages 17–27.
- Tarski, A. (1948). *A Decision Method for Elementary Algebra and Geometry*. University of California Press.
- TERA-project (1993). <http://tera.medicis.polytechnique.fr/index.html>.
- van den Dries, L. (1998). *Tame Topology and O-minimal Structures*, volume 248 of *London Mathematical Society Lecture Note Series*. Cambridge University Press.
- Vandeurzen, L., Gyssens, M., and Van Gucht, D. (1996). On query languages for linear queries definable with polynomial constraints. In Freuder, E. F., editor, *Proceedings of the 2nd Conference on Principles and practice of constraint programming*, volume 1118 of *Lecture Notes in Computer Science*, pages 468–481, Berlin. Springer-Verlag.

# Index

- Approximation of a semi-algebraic set, 36–37
- Attribute
  - geometric, 3
- CAD, 20–22, 32–33
- Cell decomposition theorem, 20
- Cell, 20
- Codd, 1
- Collins, 32
- Complex, 27
- Computationally complete query language, 57, 59
- Cone radius, 24, 45
- Cone type, 24, 26
- Cone, 23, 43
- Conical structure
  - local, 23–24, 45
- Connectivity
  - graph, 34
  - topological, 37, 52–53
- Constraint
  - linear, 4, 7, 13
  - polynomial, 5, 7, 9
- Curve selection theorem, 19
- Cut transformation, 39–40, 42
- Cylindrical algebraic decomposition, 20–22, 32–33
  - sample points, 32
  - sign conditions, 32
  - basis phase, 32
  - extension phase, 33
  - projection phase, 32
  - sector, 33
  - stack, 33
- Data
  - spatio-temporal, 17
- Database
  - constraint, 1, 5, 9, 16
  - polynomial constraint, 9
  - relation, 1
  - relational, 1, 13, 48
  - isotopic, 39
- Decomposition
  - cylindrical algebraic, 20–22, 33
  - finite cell, 21
  - topological, 19–20
- Dichotomy theorem, 35–37
- EF-game, 35, 48
- Expressiveness
  - first-order, 35
  - on finite relations over the reals, 35
  - on infinite relations, 35
- Field of real numbers, 7
- Finite structures over the reals, 36
- First-order logic
  - over the reals, 7
  - with linear constraints, 7, 13–14, 29
  - with operators, 48
  - with polynomial constraints, 7, 9, 11, 31
  - with topological operators, 60
  - with transitive closure, 48–49
  - with while-loop, 48, 57
- Formula, 7
  - atomic, 7
  - first-order, 7
  - order-generic, 36
  - quantifier-free, 7–8, 13, 17
  - equivalence, 29
  - prenex formal form, 32
- Fourier, 29
- Generic collapse, 35–36
- Geographic information system, 13–14
- Geometry
  - o-minimal, 16
  - real algebraic, 16
- GIS, 13–14
- Glue transformation, 39–40, 42
- Heintz, 34
- Hilbert, 38
- Hilbert’s 10th problem, 38
- Homeomorphism, 20, 27, 37, 54, 60
- Instance
  - database, 9, 13–14
  - relation, 14
- Isotopy, 37
- Kronecker, 34
- Local conical structure, 24
- Logic

- two-sorted, 14
- Majority, 35
- Motzkin, 29
- Number theory, 38–39
- Parity, 35
- Point structure, 26, 42
- Point
  - regular, 26
  - singular, 26
- Polynomial, 7
  - linear, 17
  - multivariate, 17
- Property
  - uniform finiteness, 37
- QEPCAD, 32
- Quantifier elimination, 8
  - by cylindrical algebraic decomposition, 32
  - first-order logic with linear constraints, 29
  - Fourier-Motzkin algorithm, 29
  - Tarski's algorithm, 31
  - first-order logic with polynomial constraints, 31
- Query language, 9–11, 14
  - closed, 11
  - closure property, 28
  - computationally complete, 57, 59
  - expressiveness, 34
  - first-order, 2, 5
- Query, 10
  - composition, 28
  - consistent, 11
  - constraint, 11
  - emptiness test, 28
  - expressibility, 11
  - membership test, 28
  - plug-in evaluation, 27
  - topological, 13, 37–38, 43, 59–60
  - unrestricted, 11
- Raster model, 14
- Real root counting, 31
- Relation, 1, 9
  - attribute, 1
  - constraint, 9
  - database, 13
  - equivalent, 10
  - finite, 6
  - finitely representable, 6
  - linear, 13
  - polynomial constraint, 9
  - semantics of a, 9
- Satisfaction, 7
- Schema
  - database, 1, 9, 13–14
- Seidenberg, 18
- Semantics
  - non-terminating, 50
  - of a linear tuple, 14
  - of a program, 58
  - of a relation, 9
  - of a transitive closure formula, 50, 55–56
  - terminating, 50
- Semi-algebraic function, 19, 21
- Semi-algebraic set, 5, 9–10, 16–20, 22–24, 27, 45, 50, 53–54, 60
  - approximation of a, 36, 54
  - decomposition, 19
  - linear approximation, 36–37
- Semi-linear set, 4, 9, 14, 16–17, 22, 47, 52, 57
- Sentence, 7
- Simplex, 27
- Sturm, 31
- Tangent line, 45
- Tarski, 18, 31
- Tarski-Seidenberg theorem, 18
- TERA project, 34
- Term, 7, 15
- Theorem
  - cell decomposition, 20
  - curve selection, 19
  - dichotomy, 35–37
  - finite cell decomposition, 21
  - Tarski-Seidenberg, 18
  - triangulation, 27, 54
  - triviality, 23–24, 60–61
- Topological property, 60
- Topological query, 37–38, 43, 59
- Topological type, 46
- Transformation
  - cut, 39–40, 42
  - glue, 39–40, 42
- Transitive closure logic, 48–49, 52
  - with start point and parameters, 56
  - with stop conditions, 55
- Transitive closure, 48, 52
- Triangulation theorem, 27, 54
- Triangulation, 27
- Trivial, 23
  - semi-algebraically, 23
- Triviality theorem, 23–24, 46, 60–61
- Triviality, 22
- Trivialization, 23
- Tuple, 1
  - linear, 13
- Turing computable, 57
- Uniform finiteness property, 21–22, 37
- Uniformly finite, 22
- Variable
  - free, 7
- Vector model, 14
- Vocabulary, 7