Chapter 13

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 (Codd, 1970). A relational database can be viewed as a finite collection of tables or relations that each contain a finite number of tuples.

Fig. 13.1 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 the relational database of Fig. 13.1 could be written as **Beer**(*Name*, *Pub*, *City*, *Postal code*), **Pub**(*Pub*, x, y).

M. Aiello, I. Pratt-Hartmann and J. van Benthem (eds.), Handbook of Spatial Logics, 799–856.

Beer

Name	*Pub*	*City*	*Postal code*
Duvel	De Muze	Antwerpen	2000
Hoegaarden	Villicus	Hasselt	3500
Geuze	La Bécasse	Brussel	1000
...	...	...	...

Pub

Pub	x	y
De Muze	16	10
Villicus	16.1	14
La Bécasse	10.4	12.3
...	...	...

Figure 13.1. An example of a relational database consisting of the two relations **Beer** and **Pub**.

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 Fig. 13.1, 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 Fig. 13.2 and he/she typically wants to ask questions that combine spatial and alpha-numeric 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 science it is customary to find *finite* representations of even infinite sets or objects.

In the 1980s, extensions of the relational model were 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. Since then, spatial database theory and technology has developed

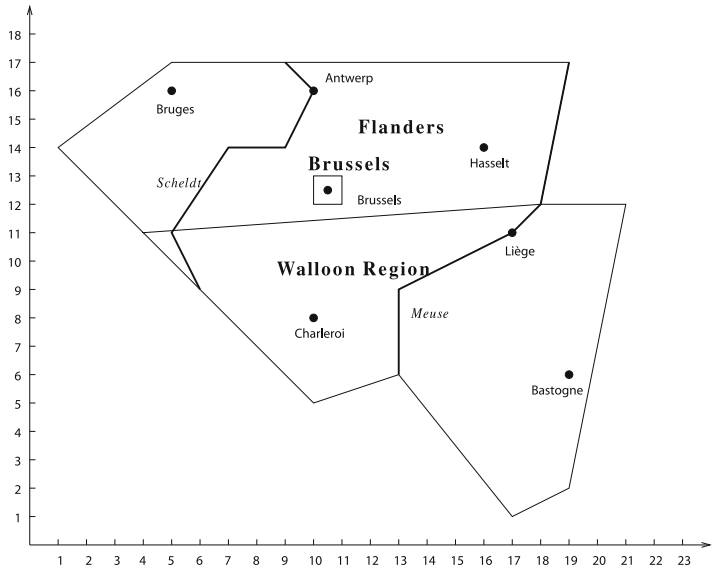

Figure 13.2. Spatial information map of Belgium.

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 Rigaux et al., 2000.

Looking again at the polylines and polygons in Fig. 13.2, 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). Moreover, 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 interpretation. 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 Fig. 13.2, is given in Fig. 13.3.

Cities

Name	*Geometry*(x, y)
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

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

Regions

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

Figure 13.3. Representation of the spatial database of Belgium shown in Fig. 13.2.

The geometric components of the relations in Fig. 13.3 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 (Rigaux et al., 2000). Indeed, in GIS, information is mostly linear in 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. Fig. 13.4 gives an example of a figure in the plane that can be described by the following combination of polynomial (in)equalities:

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

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

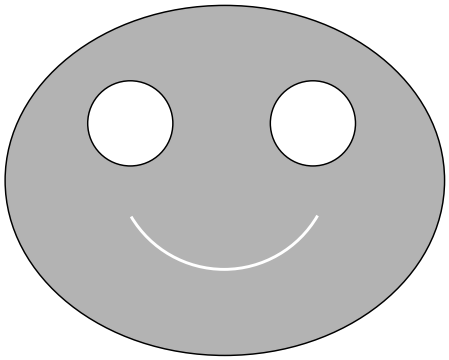

Figure 13.4. An example of a semi-algebraic set in $\mathbb{R}^2$.

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 (Bochnak et al., 1998).

Semi-algebraic sets are, together with classical alpha-numeric data, the basic ingredients in *constraint databases*. As we have seen above in Fig. 13.2, 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.

Like 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 are basically 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

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

Here, we translate "not to 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 can also 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 et al., 1995. It has received a lot of research attention since. An overview of research results in this field can be found in Kuper et al., 2000, and Revesz has written a textbook on the subject (Revesz, 2002).

Overview. This chapter is organized as follows. In Sec. 2, we describe the constraint database model with its data models and basic query languages. Sec. 3 gives an overview of some definitions and results in real algebraic geometry that will be used further on. In Sec. 4, we discuss query evaluation in the constraint database model through quantifier elimination. We also outline some quantifier elimination algorithms there. Sec. 5 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 Sec. 6, 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 $\mathrm{FO}(+, \times, <, 0, 1)$, i.e., *first-order logic with polynomial constraints*, and $\mathrm{FO}(+, <, 0, 1)$, 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 $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$ to query languages for the respective models.

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

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 $\mathrm{FO}(+, \times, <, 0, 1)$ as the *first-order logic over the vocabulary* $(+, \times, <, 0, 1)$. We build formulas in $\mathrm{FO}(+, \times, <, 0, 1)$ in the standard way: a *term* t in $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ are formulas of the form $t = t'$ or $t < t'$ for terms t and t'. Finally, formulas in $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\varphi(x_1, \ldots, x_n)$ the fact that the $\mathrm{FO}(+, \times, <, 0, 1)$ formula φ has n free variables $x_1, \ldots, x_n$. A formula without any free variables is called a *sentence*. A formula without quantifiers is called *quantifier-free*.

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

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

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

inductively on the structure of φ:

- $(\mathbb{R}, +, \times, <, 0, 1) \models (t = t')(r_1, \ldots, r_n)$ if $t(r_1, \ldots, r_n) = t'(r_1, \ldots, r_n)$;
- $(\mathbb{R}, +, \times, <, 0, 1) \models (t < t')(r_1, \ldots, r_n)$ if $t(r_1, \ldots, r_n) < t'(r_1, \ldots, r_n)$;

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

As described above, in the constraint model, the satisfaction of formulas in $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$ 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.

Of particular importance in the constraint model are the quantifier-free formulas in $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$. 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 Sec. 4, we show that both $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$ admit *quantifier elimination*. In short, this means that any formula in $\mathrm{FO}(+, \times, <, 0, 1)$ (respectively $\mathrm{FO}(+, <, 0, 1)$) is equivalent to a quantifier-free formula in $\mathrm{FO}(+, \times, <, 0, 1)$ over $\mathbb{R}$ (respectively in $\mathrm{FO}(+, <, 0, 1)$). Hence, we do not loose any generality by considering quantifier-free formulas only. As mentioned in the introduction, a (quantifier-free) formula represents a possibly infinite set of points. More specifically, they describe sets of points which correspond to *semi-algebraic sets*, in case of $\mathrm{FO}(+, \times, <, 0, 1)$, and *semi-linear sets*, in case of $\mathrm{FO}(+, <, 0, 1)$(see also Sec. 3).

EXAMPLE 13.1 In Fig. 13.4 of Sec. 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 φ has two free variables and that it uses polynomials of degree at most two.

Apart from the modeling of spatial data, the logics $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$. In the next section, we elaborate on the linear constraint model which uses $\mathrm{FO}(+, <, 0, 1)$.

The polynomial constraint data model. A *database schema* $\mathcal{S}$ is a finite set $\{S_1, \ldots, S_k\}$ of relation names. Each relation name S_i $(i = 1, \ldots, k)$ 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 $\varphi_{S_i}(x_1, \ldots, x_{n_i})$ with n_i free variables in the logic $\mathrm{FO}(+, \times, <, 0, 1)$. A *(polynomial) constraint database instance over* $\mathcal{S}$ consists of a set of constraint relations of $S_1, \ldots, S_k$.

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

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

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 13.2 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 13.1. The semantics of D is the smiling face shown in Fig. 13.4.

It is clear that the same semi-algebraic set can be represented by different formulas. Indeed, consider again Fig. 13.4. Suppose that the description of the smiling face given in Example 13.1 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 13.3 In the remainder of this chapter, we use the terms *constraint relation* and *semi-algebraic set* interchangeably, since these notions refer to the same objects, albeit from different perspectives.

Database queries in the constraint model. Before we explain how to use $\mathrm{FO}(+, \times, <, 0, 1)$ 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, as in 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 $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$-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 13.4 Let $\mathcal{S}$ consist of binary relation S. Consider the constraint query Q which maps any constraint relation of S, given by φ_S, to the highest degree of polynomials appearing in φ_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 φ_S, whereas it returns 4 on input φ'_S.

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

The logic $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, <, 0, 1)$ with predicates to address constraint relations that appear in the input database.

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

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

We also write the above for short as

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

It is clear that equivalent databases result in the same answer set. We say that φ *expresses* the corresponding (unique) unrestricted query. In the sequel, we refer to these extensions of $\mathrm{FO}(+, \times, <, 0, 1)$ simply by $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ to be closed it should be the case that the result is a quantifier-free formula in $\mathrm{FO}(+, \times, <, 0, 1)$ again. However, since $\mathrm{FO}(+, \times, <, 0, 1)$ admits quantifier-elimination, and given the way $\mathrm{FO}(+, \times, <, 0, 1)$-formulas are evaluated, this requirement is satisfied (see also Sec. 4).

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

EXAMPLE 13.5 Let $Q_{bounded}$ 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 $Q_{bounded}$.

EXAMPLE 13.6 Let $Q_{interior}$ 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, $Q_{interior}$ 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 $\mathbb{R}^2$.

In the next section, we discuss the expressive power of the query language $\mathrm{FO}(+, \times, <, 0, 1)$. 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 Sec. 5 that important queries are not expressible in $\mathrm{FO}(+, \times, <, 0, 1)$. 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 $\mathrm{FO}(+, \times, <, 0, 1)$ to richer query languages exist, some of which we discuss in Sec. 6.

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 Sec. 5), 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 $S_1, \ldots, S_k$. Each relation name S_i ($i = 1, \ldots, k$) 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

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

where $c_1, \ldots, c_{n_i}$ 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 $\varphi(x_1, \ldots, x_{m_i})$ is a quantifier-free formula in the logic $\mathrm{FO}(+, <, 0, 1)$ with m_i free variables.

The semantics of a linear tuple $t = (c_1, \ldots, c_{n_i}, \varphi(x_1, \ldots, x_{m_i}))$ 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 $\{(c_1, \ldots, c_{n_i})\} \times A_i$, in which $A_i \subseteq \mathbb{R}^{m_i}$ is the semi-linear set

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

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 $\mathcal{S} = \{S_1, \ldots, S_k\}$ of D.

For GIS, where spatial information is often modeled in either the *vector model* or the *raster model*, and combined with traditional alpha-numeric information often stored in a relational database, the linear constraint model is powerful enough. Indeed, in the vector model, three types of planar spatial objects are standardly 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 13.7 In Sec. 1, we introduced the example of a map containing infomation about Belgium, as illustrated in Fig. 13.2. 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 (Fig. 13.3). 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 $\mathrm{FO}(+, <, 0, 1)$ 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 $\mathrm{FO}(+, <, 0, 1)$ 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 $\mathrm{FO}(+, <, 0, 1)$ in the sense of a two-sorted logic. More specifically, if we consider queries over a database input schema $\mathcal{S} = \{S_1, \ldots, S_k\}$, we have, apart from the terms, formulas and quantifications possible in $\mathrm{FO}(+, <, 0, 1)$, the following ingredients:

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

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

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

EXAMPLE 13.8 An example of a (very simple) linear spatial query on the database in Example 13.3 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:

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

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 13.9 The following $\mathrm{FO}(+, <, 0, 1)$-sentence expresses $Q_{bounded}$ (see Example 13.5):

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

EXAMPLE 13.10 Several topological properties of a semi-linear set can be expressed in $\mathrm{FO}(+, <, 0, 1)$. For instance, the query $Q_{interior}$ (see Example 13.6) is expressed by the $\mathrm{FO}(+, <, 0, 1)$ formula

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

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

In spite of all this, $\mathrm{FO}(+, <, 0, 1)$ cannot be considered as a fully adequate query language for practical purposes. More specifically, there are very simple queries which are not expressible in $\mathrm{FO}(+, <, 0, 1)$, which are expressible in $\mathrm{FO}(+, \times, <, 0, 1)$. We return to this issue in Sec. 5.

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 (Coste, 2000b). Proofs of all the theorems given in this section can be found there. More advanced is the standard book in the field (Bochnak et al., 1987) and for a more algorithmic point of view we refer to (Basu et al., 2003a). An interesting book covering many other aspects of real algebraic geometry is (Benedetti and Risler, 1990). 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 (Andradas et al., 1996).

Finally, the generalization of real-algebraic geometry to so-called o-minimal geometry is described in (Coste, 2000a). An excellent book on o-minimal structures is (van den Dries, 1998). 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 (Kuper et al., 2000).

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 $\vec{x} = (x_1, \ldots, x_n)$ 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 polynomial 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

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

where $f, g_1, \ldots, g_\ell$ are multivariate polynomials in the variables $x_1, \ldots, x_n$ 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 13.11 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 $\mathrm{FO}(+, \times, <, 0, 1)$. We therefore are free to choose either of the two representations. We more often use the representation in terms of quantifier-free formulas.

EXAMPLE 13.12 In the introductory section we have already given an example of semi-algebraic sets (see, e.g., Fig. 13.4). Semi-algebraic sets can be used to model various spatial situations, but also spatio-temporal phenomena, as is illustrated in Fig. 13.5. 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

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

in which t does not appear. A fired photon torpedo follows the dotted line (between the moments $t = 0$ and $t = 1$) an then explodes (depicted as increasing dotted circles, between $t = 1$ and $t = 2$). At the bottom of Fig. 13.5 three frames of the movie are shown: at $t = 1/2, 1$ and 2. The complete movie can be described by the set

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

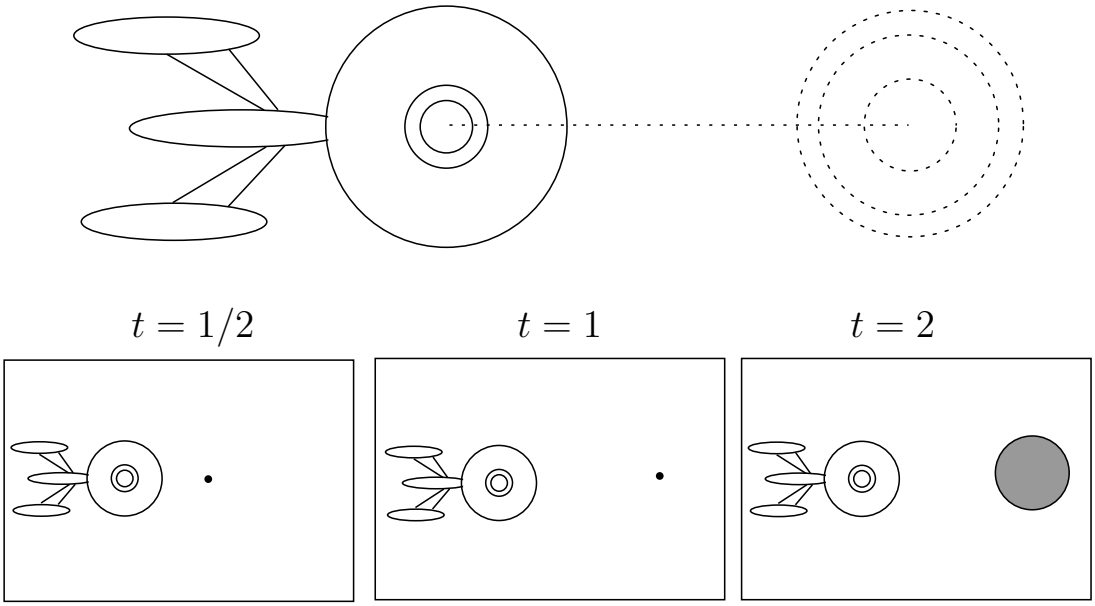

Figure 13.5. USS Enterprise firing a photon torpedo at a (cloaked) Klingon vessel.

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

THEOREM 13.13 (TARSKI-SEIDENBERG) *Let A be a semi-algebraic subset of $\mathbb{R}^{n+1}$ and let $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n$ 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. Fig. 13.6 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 13.6. 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 $\mathbb{R}$. Similarly, it is easily verified that the zig-zag line in $\mathbb{R}^2$ shown in Fig. 13.7 is not semi-algebraic.

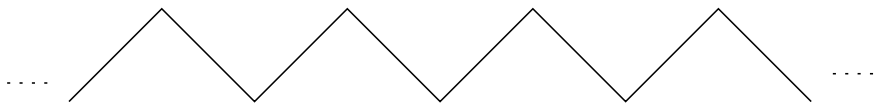

Figure 13.7. An example of a subset of $\mathbb{R}^2$ that is not semi-algebraic.

Let A be a semi-algebraic set of $\mathbb{R}^n$ and let $f : A \to \mathbb{R}$ 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 13.14 (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 $\gamma : [0,1] \to \mathbb{R}^n$ such that $\gamma(0) = \vec{x}$ and $\gamma((0,1]) \subseteq A$.*

A proof of this theorem can be found, e.g., in (Bochnak et al., 1987, Proposition 2.5.3).

REMARK 13.15 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 $\overline{U} \cap V = \emptyset$. A set A is semi-algebraically arc-connected if between any two points $\vec{s}, \vec{t} \in A$ there exists a semi-algebraic funtion $\gamma : [0,1] \to \mathbb{R}^n$ such that $\gamma(0) = \vec{s}$, $\gamma(1) = \vec{t}$ and $\gamma([0,1]) \subseteq A$. It can be easily verified that from the curve selection theorem, it follows that for a semi-algebraic A of $\mathbb{R}^n$ being connected coincides with being semi-algebraically arc-connected.

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 (Bochnak et al., 1987, Theorem 2.3.6). We first recall the definition of a homeomorphism: a *homeomorphism* h between two sets X and Y is a continuous bijection which has continuous inverse. Two sets are called *homeomorphic* if there exists a homeomorphism between them.

THEOREM 13.16 *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.

Cylindrical algebraic decomposition. In practice, more refined decompositions of semi-algebraic sets are used that are also 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

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

of open intervals and points, where $a_1 < \cdots < a_k$ 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, $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is again the usual projection map defined by $\pi(x_1, \ldots, x_n, x_{n+1}) = (x_1, \ldots, x_n)$.

We still have to specify what a cell in $\mathbb{R}^{n+1}$ is. Let $(i_1, \ldots, i_m)$ 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 $(i_1, \ldots, i_m)$-cells are already defined. Then an $(i_1, \ldots, i_m, 0)$-cell is the graph $\Gamma(f)$ of a continuous semi-algebraic function $f{:}X{\to}\mathbb{R}$, where X is an $(i_1, \ldots, i_m)$-cell. Furthermore, an $(i_1, \ldots, i_m, 1)$-cell is a set of the form

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

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

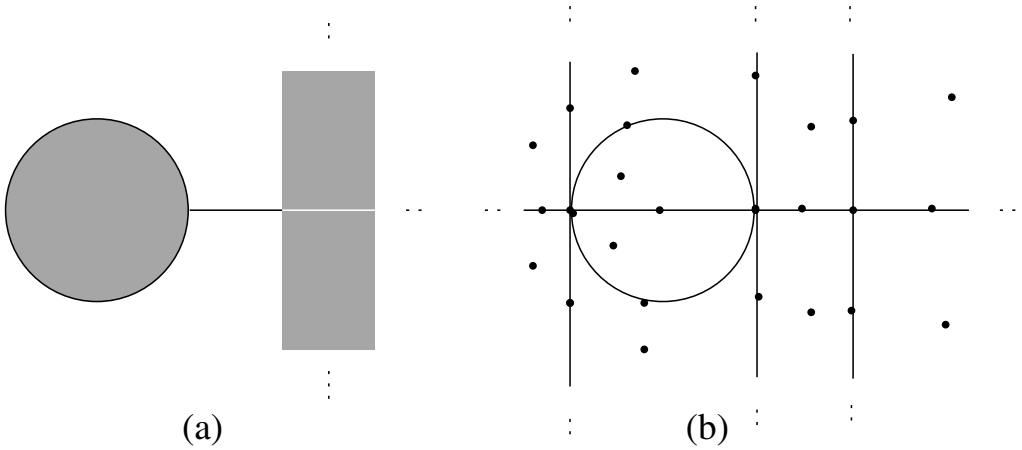

Figure 13.8. An example of a CAD in $\mathbb{R}^2$.

A *cell in* $\mathbb{R}^n$ is an $(i_1, \ldots, i_n)$-cell for some sequence $(i_1, \ldots, i_n)$. 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 13.17 (FINITE CELL DECOMPOSITION) *Given any semi-algebraic sets $A_1, \ldots, A_k$ of $\mathbb{R}^n$, there is a CAD of $\mathbb{R}^n$ partitioning each $A_1, \ldots, A_k$.* QED

EXAMPLE 13.18 Consider the semi-algebraic subset of $\mathbb{R}^2$ given by the formula

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

This set is shown in part (a) of Fig. 13.8. 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 (van den Dries, 1998, Ch. 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 Sec. 5. 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 finite. We call A *uniformly finite over* $\mathbb{R}^n$ if there is an $N \in \mathbb{N}$ such that $|A_{\vec{x}}| \leqslant N$ for all $\vec{x} \in \mathbb{R}^n$. We then have:

THEOREM 13.19 (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 Fig. 13.8, 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 13.20 Consider again the semi-algebraic set A and CAD of $\mathbb{R}^2$ of Example 13.18, shown in Fig. 13.8. 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 $\pi : \mathbb{R}^2 \to \mathbb{R}$ on the x-axis is *trivial* over each of the cells $C_1, \ldots, C_7$.

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 $f : A \to B$ be a continuous semi-algebraic map. 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, λ) consisting of semi-algebraic set $F \subseteq \mathbb{R}^N$, for some N, and semi-algebraic map $\lambda : A \to F$ such that $(f, \lambda) : A \to B \times F$ is a homeomorphism.

Let $f : A \to B$ and suppose that f has a semi-algebraic trivialization. Then it is easy to show that all fibres are semi-algebraically 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 13.21 (TRIVIALITY THEOREM) *Let $f : A \to B$ be a continuous semi-algebraic map as above. Then there is a finite partition of $B = B_1 \cup \cdots \cup B_\ell$ such that each B_i is semi-algebraic and f is semi-algebraically trivial over each B_i.*

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 ε and let $S^{n-1}(\vec{p},\varepsilon)$ be the sphere with center $\vec{p}$ and radius ε.

We denote by $\mathrm{Cone}\big(\vec{p}, S^{n-1}(\vec{p},\varepsilon)\cap A\big)$ 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 $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ denote the standard Euclidean norm.

THEOREM 13.22 (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*

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

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

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

The local conical structure theorem is a direct consequence of the triviality theorem, for $f : A\to\mathbb{R}$ defined as $f(\vec{x}) = \|\vec{x}-\vec{p}\|$.

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 Fig. 13.9. For the point $\vec{p}$, we have indicated a cone radius ε 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 Fig. 13.9 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 $(0,0,2)$

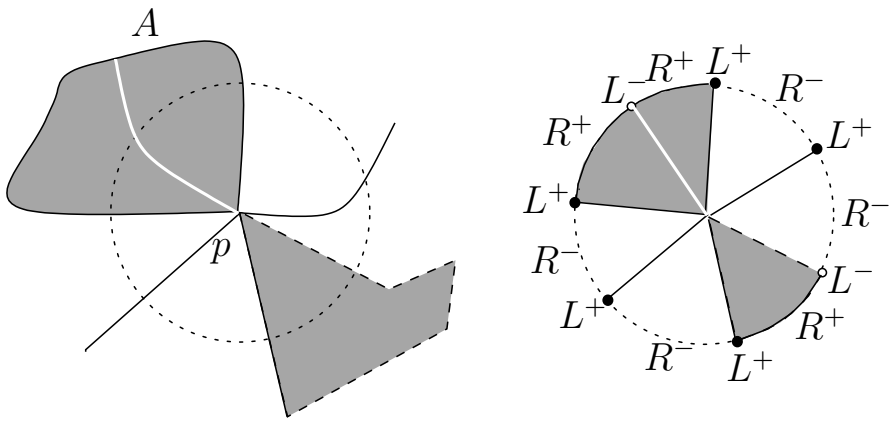

Figure 13.9. A semi-algebraic set A of $\mathbb{R}^2$ and the cone type of A in its points p given by the circular list $(L^+R^-L^+R^-L^-R^+L^+R^-L^+R^-L^+R^+L^-R^+)$.

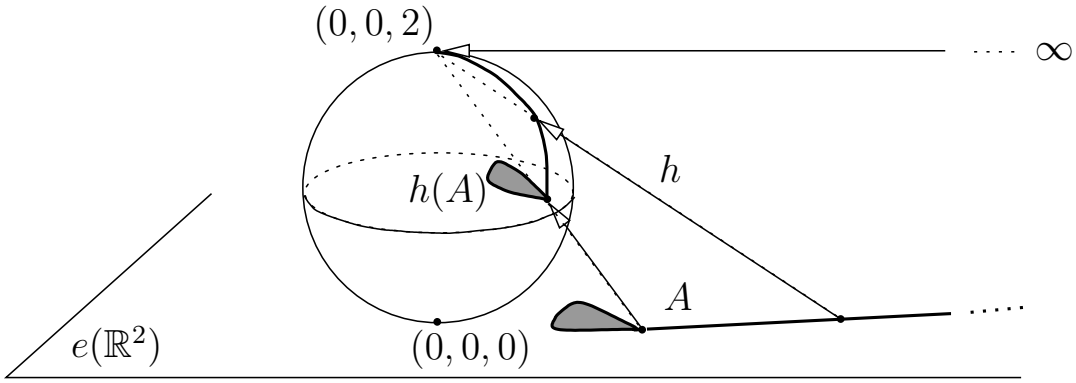

Figure 13.10. Illustration of the stereographical projection h.

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 ∞ and base $S^1((0,0),\varepsilon) \cap A$.

More formally, consider the embedding e of $\mathbb{R}^2$ in $\mathbb{R}^3$ that maps (x,y) to $(x,y,0)$. Let σ be the reflection of $\mathbb{R}^3$ defined by $(x,y,z) \mapsto (x,y,2-z)$. Finally, let $h : e(\mathbb{R}^2) \cup \{\infty\} \to S^2((0,0,1),1)$ 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 the cone type of A in ∞ to be the cone type of the point $(0,0)$ in the set $e^{-1}(h^{-1}(\sigma(\{(0,0,2)\} \cup h(e(A))) \setminus \{\infty\}))$. We remark that the cone type of A in ∞ is (E) if and only if A is a bounded subset of $\mathbb{R}^2$.

Let A be a semi-algebraic set in $\mathbb{R}^2$ and let $\vec{p}$ be a point in the closure of A. Then it easily verified that for any two cone radii ε_1 and ε_2 of A in $\vec{p}$ (and ∞)

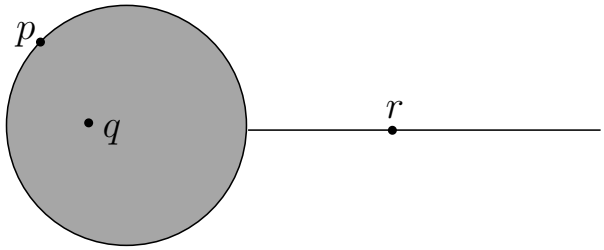

Figure 13.11. Types of regular points of a closed semi-algebraic set: $\Pi(p) = (R^- L^+ R^+ L^+)$, $\Pi(q) = F$ and $\Pi(r) = (R^- L^+ R^- L^+)$.

we get the same cone type. In other words, the notion of the cone type of A in $\vec{p}$ (and ∞) is well-defined.

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

DEFINITION 13.23 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 13.24 *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*. It can be shown that only a finite number of cone types appear in A. Moreover, 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 13.25 Suppose that A is a closed semi-algebraic set of $\mathbb{R}^2$. By the the observation above, 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). Fig. 13.11 illustrates these different cone types.

In Sec. 5, 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 $a_0, a_1, \ldots, a_k$ be $(k+1)$ affine independent points in $\mathbb{R}^n$. A *k-simplex* $(a_0, a_1, \ldots, a_k)$ is the set of points

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

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

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

A *face of* σ is a simplex corresponding to any nonempty subset of $(a_0, a_1, \ldots, a_k)$. 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 $\mathrm{cl}(\sigma_1) \cap \mathrm{cl}(\sigma_2) = \emptyset$, or $\mathrm{cl}(\sigma_1) \cap \mathrm{cl}(\sigma_2) = \mathrm{cl}(\tau)$, where τ is a common face of σ_1 and σ_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 13.26 *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 $\mathrm{FO}(+, <, 0, 1)$ and $\mathrm{FO}(+, \times, <, 0, 1)$, may be evaluated.

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

- we plug-in the descriptions $\varphi_{S_1}(x_1, \ldots, x_{n_1}), \ldots, \varphi_{S_k}(x_1, \ldots, x_{n_k})$ of the input relations into the query formula $\varphi(x_1, \ldots, x_m)$ (this means that we replace each occurrence of some $S_i(v_1, \ldots, v_{n_i})$ in the query formula by $\varphi_{S_i}(v_1, \ldots, v_{n_i})$);
- 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 $\mathrm{FO}(+, \times, <, 0, 1)$ of the output relation.

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

EXAMPLE 13.27 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 $(x')^2 + (y')^2 \leq 4$. This gives rise to the formula

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

The formula ψ contains three quantifiers. When we eliminate the quantifiers from ψ, 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 $\mathrm{FO}(+, \times, <, 0, 1)$-formulas of the input relations into the query formula yields a formula in $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$. 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 $\mathrm{FO}(+, <, 0, 1)$

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

THEOREM 13.28 *The theory of* $(\mathbb{R}, +, <, 0, 1)$ *admits quantifier elimination. More specifically, this means that there is an algorithm that on input of a formula* $\varphi(x_1, \ldots, x_n)$ *over* $(+, <, 0, 1)$ *returns a quantifier-free formula* $\psi(x_1, \ldots, x_n)$ *that is equivalent to* $\varphi(x_1, \ldots, x_n)$ *over* $(\mathbb{R}, +, <, 0, 1)$.

We say that two formulas $\varphi(x_1, \ldots, x_n)$ and $\psi(x_1, \ldots, x_n)$ 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 (Motzkin, 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 $\varphi(x_1, \ldots, x_{m-1})$ of the form

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

where $\psi(x_1, \ldots, x_m)$ is a Boolean combination of atomic formulas of $\mathrm{FO}(+, <, 0, 1)$. So, $\psi(x_1, \ldots, x_m)$ 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 the variables $x_1, \ldots, x_{m-1}$, the terms t_{ij} can be ordered, let us say (after re-indexing) as $t_1 \leq t_2 \leq \cdots \leq t_k$. It is clear that for any two values of x_m taken strictly between some t_i and t_{i+1} (see Fig. 13.12), 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.

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

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

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.

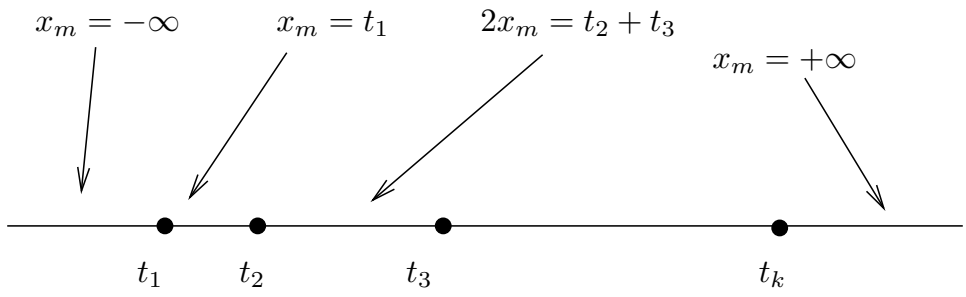

Figure 13.12. The relevant values for x_m.

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 13.29 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 $\psi(x, y) =$

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

As mentioned above, we remark that the $-\infty$ and $+\infty$ from the algorithm are implemented by substracting 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.

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

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

THEOREM 13.30 *There is an algorithm that on input of a* $\mathrm{FO}(+,\times,<,0,1)$*-formula* $\varphi(x_1,\ldots,x_n)$ *returns a quantifier-free* $\mathrm{FO}(+,\times,<,0,1)$*-formula* ψ $(x_1,\ldots,x_n)$ *that is equivalent to the given formula* $\varphi(x_1,\ldots,x_n)$ *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 13.31 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, 1954, but a major breakthrough was achieved in Collins, 1975, which 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 us 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 to be implemented, however. It has undergone numerous improvements, resulting in the implementation QEPCAD (Quantifier Elimination by Partial Cylindrical Algebraic Decomposition) by Hong, 1990. We refer to Caviness and Johnson, 1998 for a description at length of the current state of CAD.

A formal definition of a CAD was given in Sec. 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 $\mathrm{FO}(+,\times,<,0,1)$-formula in prenex normal form

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

Here, $\bar{\varphi}(u_1,\ldots,u_m,x_1,\ldots,x_n)$ 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 $(x_1, \ldots, x_n)$ and parameters $(u_1, \ldots, u_m)$ of the given formula $\varphi(u_1, \ldots, u_m, x_1, \ldots, x_n)$. The goal is to eliminate the variables from the formula $\varphi(u_1, \ldots, u_m)$ via the computation of a CAD of $\bar{\varphi}(u_1, \ldots, u_m, x_1, \ldots, x_n)$.

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

- *the projection phase*: here the $(m + n)$-dimensional semi-algebraic set A is iteratively projected onto lower dimensional spaces ($\mathbb{R}^{m+n} \to \mathbb{R}^{m+n-1} \to \cdots \to \mathbb{R}^1$);
- *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 ($\mathbb{R}^1 \to \mathbb{R}^2 \to \cdots \to \mathbb{R}^{m+n-1} \to \mathbb{R}^{m+n}$), 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 $C_1, \ldots, C_{m+n}$, where each C_i is a partition of $\mathbb{R}^i$ into cells. Each cell C is given by means of quantifier-free $\mathrm{FO}(+, \times, <, 0, 1)$-formula φ_C 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.

By construction, the set $\{(u_1, \ldots, u_m) \in \mathbb{R}^m \mid \varphi(u_1, \ldots, u_m)\}$ consists of all cells C in C_m for which there exists a cell C' in C_{m+n} which is in the stack $C \times \mathbb{R}^n$ of C and which belongs to A. Hence, a quantifier-free equivalent formula for $\varphi(u_1, \ldots, u_m)$ is obtained as a disjunction of all formulas φ_C describing cells C in C_m such that in the stack above C a cell of C_{m+n} belongs to A.

EXAMPLE 13.32 We illustrate the CAD algorithm using the three-dimensional set given by the quantifier-free $\mathrm{FO}(+, \times, <, 0, 1)$-formula

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

This set is depicted in Fig. 13.13.

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 Fig. 13.14, 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).

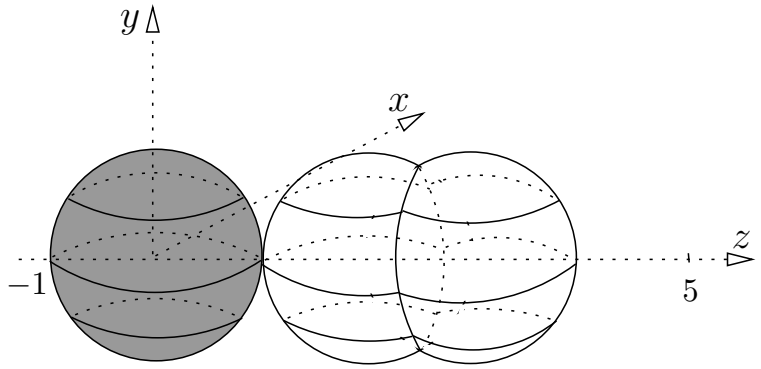

Figure 13.13. An example of a semi-algebraic set in $\mathbb{R}^3$.

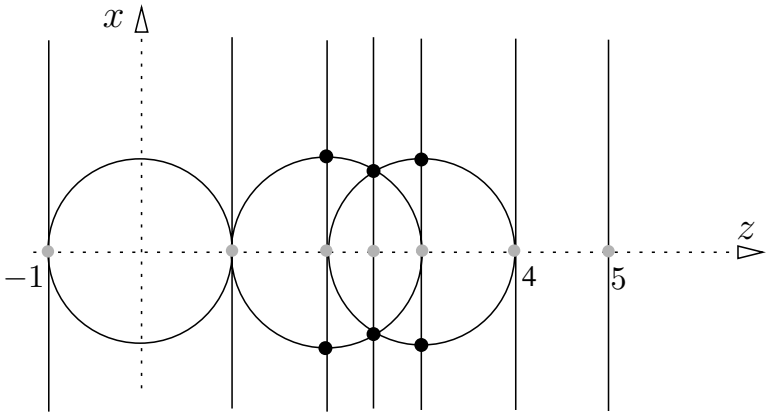

Figure 13.14. The 1- and 2-dimensional induced CADs of the semi-algebraic set of Fig. 13.13.

In the example of Fig. 13.13 and 13.14, 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 CAD. The stack above the second interval, for intance, 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 were 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 (Heintz et al., 1993). 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 (Heintz et al., 1993). In the TERA project, the software *Kronecker* was developed and it is for the moment the most efficient software for quantifier

elimination (TERA-project, 1993) over the reals. The Kronecker implementation is decribed in (Giusti et al., 2001). We refer to (Basu et al., 2003b) 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 $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ 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 (Benedikt et al., 1996). 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 (Benedikt and Libkin, 2000). 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 for two finite sets S_1 and S_2 is: $\textsc{majority}(S_1, S_2)$ is true if and only if $S_2 \subseteq S_1$ and $|S_1| \leq 2|S_2|$;
- The decision problem PARITY for a finite set S is: $\textsc{parity}(S)$ is true if and only if $|S|$ is even.

The proof of the following lemma is a routine exercise in finite-model theory (Ebbinghaus et al., 1984). It can, e.g., be proven using the well-known technique of Ehrenfeucht-Fraïssé games. This lemma holds for arbitrary finite structures.

LEMMA 13.33 *On finite structures over the signature* $(<, S_1, S_2)$*, the decision problem* MAJORITY(S_1, S_2) *is not expressible in* FO$(<, S_1, S_2)$*. Likewise, on finite structures over the signature* $(<, S)$*, the decision problem* PARITY(S) *is not expressible in* 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 (Benedikt et al., 1996). 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 (Belegradek et al., 1996; Benedikt and Libkin, 1996; Benedikt and Libkin, 1997; Grumbach and Su, 1995; Grumbach et al., 1995; Paredaens et al., 1995; Stolboushkin and Taitslin, 1996).

Consider structures over the vocabulary $(+, \times, <, 0, 1, S_1, \ldots, S_k)$ 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 $(+, \times, <, 0, 1, S_1, \ldots, S_k)$ is called *order-generic* if on such structures, it is invariant under monotone bijections $f : \mathbb{R} \to \mathbb{R}$. Benedikt, Dong, Libkin, and Wong showed the following (Benedikt et al., 1996):

THEOREM 13.34 (COLLAPSE THEOREM) *For each order-generic formula in* FO$(+, \times, <, 0, 1, S_1, \ldots, S_k)$*, there exists a formula in* FO$(<, S_1, \ldots, S_k)$*, 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* $S_1, \ldots, S_k$.

The following lemma, which specializes Lemma 13.33 from general finite ordered structures to finite structures over the reals, now follows directly from Lemma 13.33, Theorem 13.34 and the observation that the properties PARITY and MAJORITY of finite structures over the reals are invariant under monotone bijections from $\mathbb{R}$ to $\mathbb{R}$.

LEMMA 13.35 *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 useful for showing that the expressive power of FO$(+, \times, <, 0, 1)$ is rather limited on finite structures over the reals.

THEOREM 13.36 (DICHOTOMY THEOREM) *Let* φ *be a* FO$(+, \times, <, 0, 1, S_1, \ldots, S_k)$*-formula. There exists a polynomial* p_φ *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.

We provide an example of how to use the dichotomy theorem for showing inexpressibility results in the next section.

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 ε-approximation of semi-algebraic sets.

DEFINITION 13.37 Let A be a semi-algebraic set of $\mathbb{R}^2$. A *(linear) ε-approximation* of A is a semi-linear set B of $\mathbb{R}^2$ which is homeomorphic to A via a homeomorphism $h : \mathbb{R}^2 \to \mathbb{R}^2$, and such that for any $\vec{p} \in A$, $d(\vec{p}, h(\vec{p})) < \varepsilon$.

THEOREM 13.38 *Let $\varepsilon > 0$ be a real number. There is no $(+, \times, <, 0, 1, S)$-formula that expresses a linear ε-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 ε-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 $\mathrm{FO}(+, \times, <, 0, 1)$(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 ε-approximation query can be expressed $\mathrm{FO}(+, \times, <, 0, 1, S)$, then Q is also expressible in $\mathrm{FO}(+, \times, <, 0, 1, S)$ by a formula φ. 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_φ such that the output of φ, when applied to D is bounded by $p_\varphi(|D|) = p_\varphi(3)$. QED

We remark that an alternative proof of Theorem 13.38 can be obtained 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 $\mathrm{FO}(+, \times, <, 0, 1)$. First, we remark that topological connectivity of a set in the plane is a query that is invariant under topological

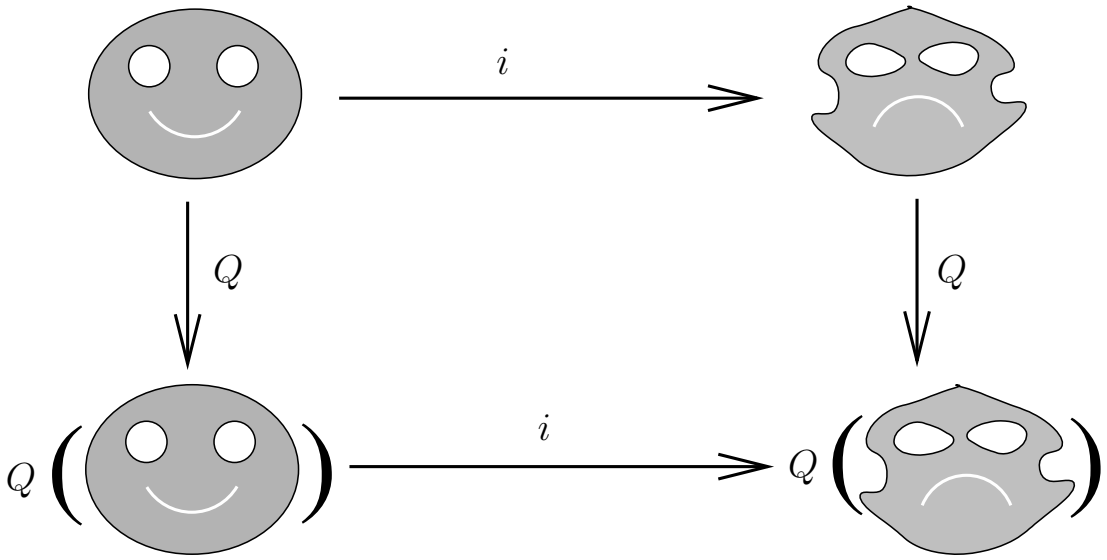

Figure 13.15. The query Q is topological.

transformations of the plane, i.e., it is invariant under isotopies $i : \mathbb{R}^2 \to \mathbb{R}^2$ (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 $i : \mathbb{R}^2 \to \mathbb{R}^2$ such that $i(A) = B$, $i(Q(A)) = Q(B)$ holds. This concept is illustrated in Fig. 13.15

We remark that deciding whether a query is topological is undecidable. For completeness we give the proof first presented in Paredaens et al., 1994.

THEOREM 13.39 *Testing whether a query expressible in* $\mathrm{FO}(+, \times, <, 0, 1)$ *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

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

$$\mathrm{enc}(n_1, n_2, \ldots, n_k) = \mathrm{enc}(n_1) \cup (\mathrm{enc}(n_2) + n_1 + 2) \cup \cdots \cup (\mathrm{enc}(n_k) + n_1 + 2 + \cdots + n_{k-1} + 2).$$

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

We now associate with each first-order sentence

$$\forall x_1 \forall x_2 \cdots \forall x_n \varphi(x_1, \ldots, x_n)$$

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

$$\begin{aligned} Q_\varphi \quad = \quad & \textbf{if } S \text{ encodes a vector } (n_1, \ldots, n_k) \in \mathbb{N}^k \textbf{ then} \\ & \quad \textbf{if } \varphi(n_1, \ldots, n_k) \textbf{ then return } \emptyset \\ & \quad \textbf{else return } 0 \\ & \textbf{else return } \emptyset. \end{aligned}$$

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

In the remainder of this section we investigate which databases can be distinguished by means of topological queries expressible in $\mathrm{FO}(+, \times, <, 0, 1, S)$. $\mathrm{FO}(+, \times, <, 0, 1, S)$. 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, in general, many semi-algebraic sets, but in which only points of "regular" cone types are allowed. The latter case is discussed in (Grohe and Segoufin, 2002). We will only describe the case of closed databases. The following results are taken from (Kuijpers et al., 2000).

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 Fig. 13.16. 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 Fig. 13.16).

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 $\mathrm{FO}(+, \times, <, 0, 1, S)$.

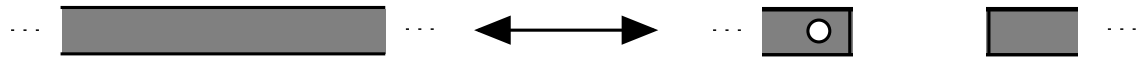

Figure 13.16. The cut and glue transformations on figures in $\mathbb{R}^2$.

THEOREM 13.40 *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 $\mathrm{FO}(+,\times,<,0,1,S)$-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 φ, which expresses a topological query, such that $\varphi(A) = \textsc{True}$ and $\varphi(B) = \textsc{False}$.

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 $\mathrm{FO}(+,\times,<,0,1,S_1,S_2)$ such that for any finite database $D = (D_1, D_2)$ over (S_1, S_2), we have that for $E_A(D) = \{(x,y) \in \mathbb{R}^2 \mid (\mathbb{R}, D) \models \psi(x,y)\}$, $\varphi(E_A(D)) = \textsc{True}$ if and only if $\textsc{majority}(D_1, D_2)$ is FALSE.

By Lemma 13.35, this then yields the desired contradiction. This reduction technique is inspired by (Grumbach and Su, 1995).

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 $D_1 = \{r_1, \ldots, r_n\}$ and $D_2 = \{a_1, \ldots, a_m\}$ with $0 < r_1 < \cdots < r_n$ and $0 < a_1 < \cdots < a_m$, we construct within the fixed rectangular part α 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 (Grumbach and Su, 1995) and is illustrated in Fig. 13.17 for $n = 6$ and $m = 4$. The construction is as follows. Take a rectangular subarea α' of α. Let (b_0, s_0) be the left bottom corner of α' and let h and w be its height and width. Then sets $D_1' = \{s_0, \ldots, s_n\}$ and $D_2' = \{b_0, b_1, \ldots, b_m, b_{m+1}, \ldots, b_{2m}\}$, 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 $i = j = 0$,

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 $0 \leq j < n$,

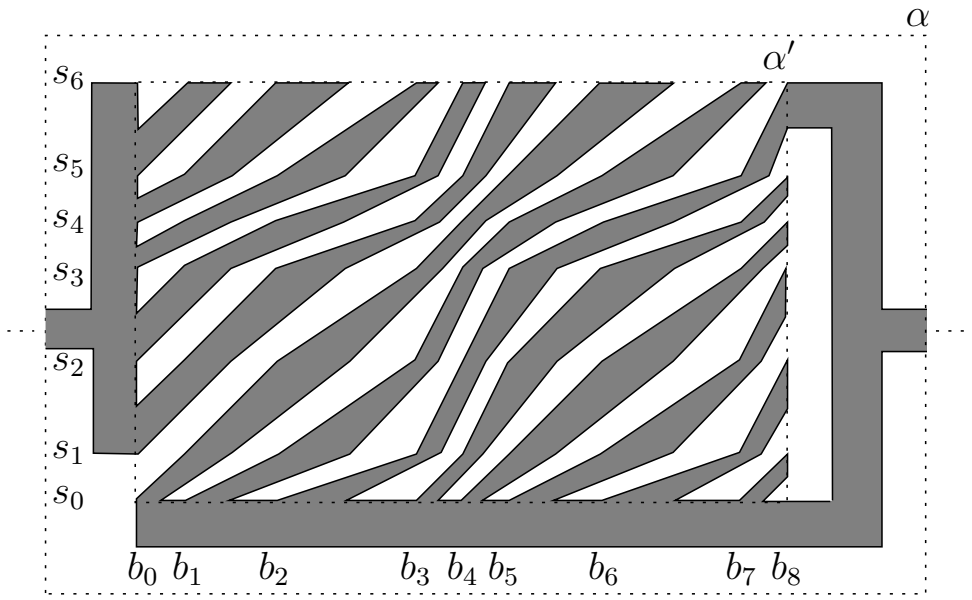

Figure 13.17. Construction of $E(D)$ for $D = (D_1, D_2)$ with $D_1 = \{1, 3, 5, 6, 7, 9\}$ and $D_2 = \{1, 3, 6, 7\}$ in the rectangular area α.

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 Fig. 13.17) 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 $\mathrm{FO}(+, \times, <, 0, 1, S_1, S_2)$.

We then glue $E(D)$ to the part of A outside the cut-or-glue transformation area α 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 ψ in $\mathrm{FO}(+, \times, <, 0, 1, S_1, S_2)$ 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 Fig. 13.16 if $\textsc{majority}(D_1, D_2)$ is true, and homeomorphic to the left part of Fig. 13.16 otherwise.

Hence, in case of majority, $E_A(D)$ is homeomorphic to B, and in the other case it is homeomorphic to A. Since φ expresses a topological query which distinguishes between A and B, we can use φ to express MAJORITY. QED

With some more work additional cut and glue transformations can be proved to produce results that are indistinguishable from the original spatial figure. Three such transformations are shown in Fig. 13.18. In (a), we have a stip that can be cut (this time without producing a perforation in one of the sides). In

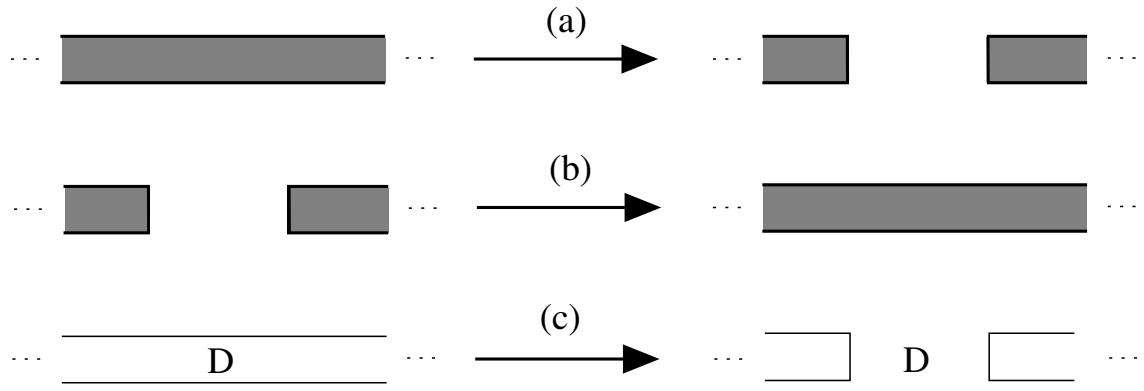

Figure 13.18. Three more cut and glue transformations on figures in $\mathbb{R}^2$.

Figure 13.19. One and two balls cannot be distinguished by a topological $\mathrm{FO}(+, \times, <, 0, 1)$-query.

(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.

REMARK 13.41 A direct consequence of Theorem 13.40 is that the connectivity query is not expressible in $\mathrm{FO}(+, \times, <, 0, 1)$. Indeed, this follows immediately from the observation that two disks can be transformed into a single disk using a cut and glue transformation (see Fig. 13.19).

We remark that Theorem 13.40 also holds when semi-algebraic is replaced by semi-linear. In this case, linear versions of the cut and glue transformations have to be considered.

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 $\mathrm{FO}(+, \times, <, 0, 1)$-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 arises as to 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 13.23 of the point structure of a database.

DEFINITION 13.42 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 13.43 *Let A and B be closed semi-algebraic sets in $\mathbb{R}^2$. The sets A and B are indistinguishable by topological $\mathrm{FO}(+, \times, <, 0, 1)$-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 Sec. 5.4, we will discuss in more detail how a cone of a point can be expressed in $\mathrm{FO}(+, \times, <, 0, 1)$.

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 Fig. 13.18. 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 Fig. 13.20. QED

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 Fig. 13.21 can be distinguished by a topological $\mathrm{FO}(+, \times, <, 0, 1)$-sentence even though they have isomorphic cone structures (Grohe and Segoufin, 2002).

Theorem 13.43 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 $\mathrm{FO}(+, \times, <, 0, 1)$. There are results that characterize this expressive power, however. It has been shown that the topological fragment of $\mathrm{FO}(+, \times, <, 0, 1)$ just allows us to formulate queries that talk about types of cones appearing in a semi-algebraic sets and on the number of points having particular cones (Benedikt et al., 2006).

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

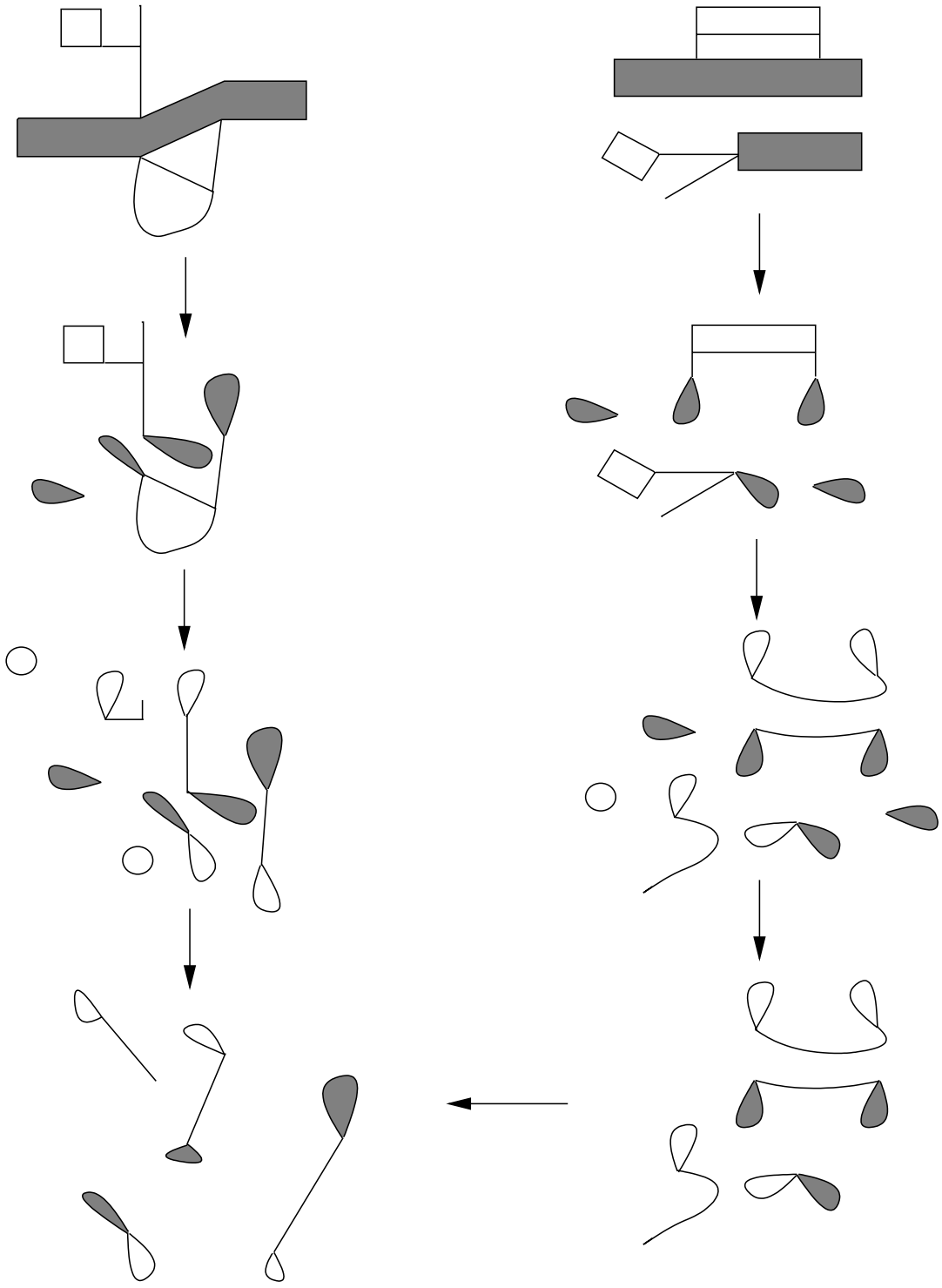

Figure 13.20. The transformation of two closed semi-algebaric sets (on the left and right hand top) into their canonical form (left bottom).

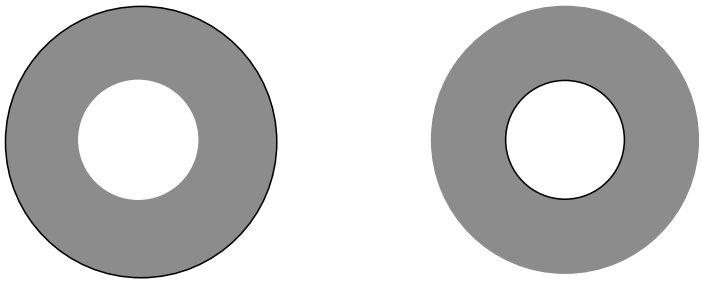

Figure 13.21. Semi-open annuli with the opposite open sides can be distinguished by a topological $\mathrm{FO}(+, \times, <, 0, 1)$-query.

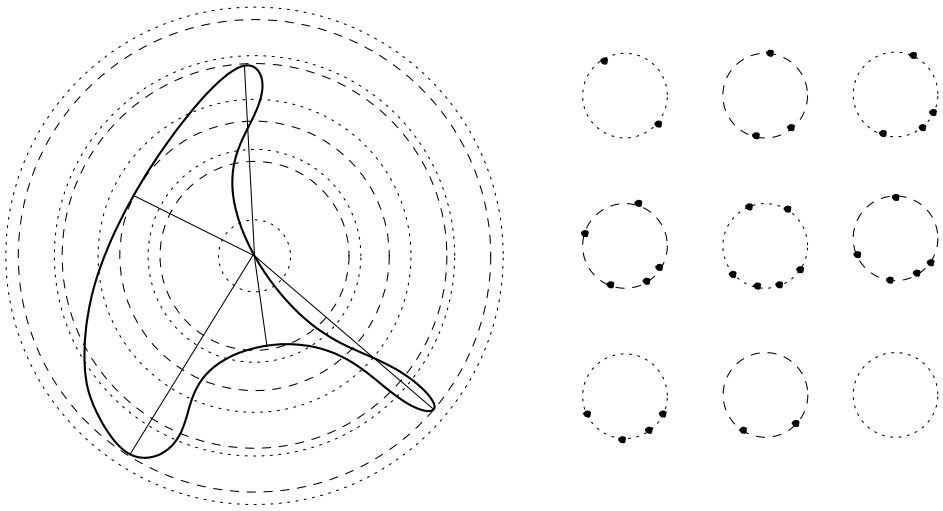

Figure 13.22. Illustration of how the intersections $S^1(\vec{p}, r) \cap A$ change in terms of r.

properties. In this section, we will show that $\mathrm{FO}(+, \times, <, 0, 1)$ is expressive enough to find a cone radius for each point in the database.

More specifically, the following result holds (Geerts, 2003).

THEOREM 13.44 *There exists an* $\mathrm{FO}(+, \times, <, 0, 1, S_1, S_2)$*-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 afield. 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 Fig. 13.22. 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 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 an 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 $\vec{q} - \vec{p}$.

It can be shown that there exists an $\mathrm{FO}(+, \times, <, 0, 1, S_1, S_2)$-formula φ which returns for any pair $(A, \vec{p})$ all point $\vec{q}$ which have a tangent line perpendicular to the vector $\vec{q} - \vec{p}$. Using φ we 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 interesections $S^1(\vec{p}, r) \cap A$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ and $\mathrm{FO}(+, <, 0, 1)$

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

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

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

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 $\mathrm{FO}(+, \times, <, 0, 1)$ 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 (Vandeurzen et al., 1996).

PROPOSITION 13.45 *A semi-linear set of $\mathbb{R}^n$ is convex if and only if it is closed under taking midpoints.*

We remark this property does not hold for subsets of $\mathbb{R}^n$ that are not semi-linear, e.g., think of $\mathbb{Q}^n$.

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

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

It is not true, however, that all queries expressible in $\mathrm{FO}(+, \times, <, 0, 1)$ are also expressible in $\mathrm{FO}(+, <, 0, 1)$. Indeed, $\mathrm{FO}(+, \times, <, 0, 1)$ is clearly more expressive than $\mathrm{FO}(+, <, 0, 1)$ as far as queries that return some n-dimensional result are concerned. For example, the constant query that returns on any input the n-dimensional unit sphere, for instance, is not expressible in $\mathrm{FO}(+, <, 0, 1)$. But what about Boolean queries? It turns out that $\mathrm{FO}(+, \times, <, 0, 1)$ is again more expressive than $\mathrm{FO}(+, <, 0, 1)$ as far as Boolean queries are concerned, as is illustrated by the following theorems. The first theorem is from (Afrati et al., 1994) and was proved using Ehrenfeucht-Fraïssé games.

THEOREM 13.46 *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* $\mathrm{FO}(+, \times, <, 0, 1, S)$*, but not in* $\mathrm{FO}(+, <, 0, 1, S)$.

Another example of a property that is not expressible in $\mathrm{FO}(+, <, 0, 1)$ is from (Benedikt and Keisler, 2000).

THEOREM 13.47 *The Boolean query deciding whether a semi-linear subset S of $\mathbb{R}^2$ contains a line is expressible in* $\mathrm{FO}(+, \times, <, 0, 1, S)$*, but not in* $\mathrm{FO}(+, <, 0, 1, S)$.

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 (Geerts and Kuijpers, 2000; Geerts and Kuijpers, 2005) and (Kreutzer, 2001). Next, we extend first-order logic with a while-loop (Gyssens et al., 1999). We conclude the section by extending first-order logic with specific operators for topological properties (Benedikt et al., 2003).

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 $\mathcal{S}$ consists of a binary relation name S. The transitive closure of D is usually computed in stages X_i, $i = 0, 1, 2, \ldots$. Initially, $X_0 = D$ and for $n \geqslant 0$ we have

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

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 (Ebbinghaus and Flum, 1995).

EXAMPLE 13.48 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

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

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

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

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 $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n)$ such that $D(v_i, v_{i+1})$ for $i = 0, 1, \ldots, n - 1$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$+TC.

First-order logic with transitive-closure operator. A formula in $\mathrm{FO}(+, \times, <, 0, 1)$+TC is a formula built in the same way as an $\mathrm{FO}(+, \times, <, 0, 1)$ 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 ψ among $\vec{x}$, $\vec{y}$, and if $\vec{s}$, $\vec{t}$ are k-tuples of real variables, then

$$\big[\mathrm{TC}_{\vec{x};\vec{y}}\, \psi(\vec{x}, \vec{y})\big](\vec{s}, \vec{t}) \tag{13.1}$$

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

We will distinguish between $\mathrm{FO}(+,<,0,1)+\mathrm{TC}$ and $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ 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 (13.1), we compute again the following stages

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

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

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

In general, the semantics of a formula φ in $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ 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 φ.

However, in this context, we 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_∞ might not terminate. Hence, the property of effective computability of queries expressible in $\mathrm{FO}(+,\times,<,0,1)$ or $\mathrm{FO}(+,<,0,1)$ is lost when extending these logics with recursion. In case the computation of X_∞ does not terminate, then the semantics of the formula (13.1) (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 $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$.

EXAMPLE 13.49 Let $\mathcal{S}$ consist of the binary relation S. Consider the $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ formula

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

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

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

It is clear that for all $n \geqslant 0$, $X_n \neq X_{n+1}$ and hence we need to compute infinitely many stages until the fixed point X_∞ is reached. We have depicted this example in Fig. 13.23. It easy to see t hat X_∞ is not a semi-algebraic set.

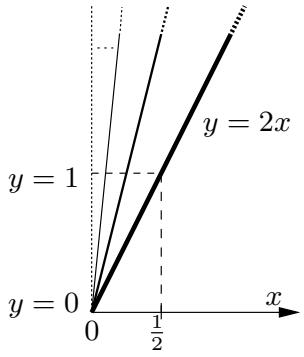

Figure 13.23. Computation of $\big[TC_{x;y}S(x,y)\big](s,t)$ for $D = \{(x,y) \mid y = 2x\}$ (thickest line). Consecutive stages are drawn in decreasingly finer lines.

On the other hand, even when X_∞ is semi-algebraic, its computation may still be non-terminating. Moreover, one may wonder whether the non-termination of the $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ formula in Example 13.49 is caused by the unboundedness of the input database. However, it easy to see that even bounded input databases may cause non-termination.

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

EXAMPLE 13.50 Let the database schema $\mathcal{S}$ consist of the unary relation name S. Consider the formula

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

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 $n \geqslant 0$,

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

It is clear that $X_{p+1} = X_p$ and hence this $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ formula is terminating. Moreover, it expresses the query Q_{nat} which tests whether the number stored in a database is a natural number. We have illustrated this example in Fig. 13.24. It is easily verified that the query Q_{nat} cannot be expressed in the logic $\mathrm{FO}(+,\times,<,0,1)$.

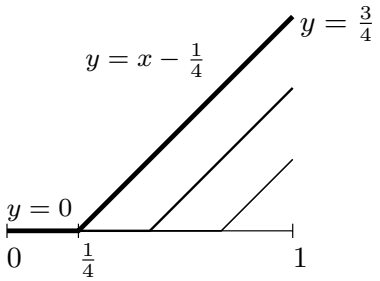

Figure 13.24. Example of the terminating $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ formula of Example 13.50 for $D = \{4\}$.

As a result of the above example and Example 13.50, we have that $\mathrm{FO}(+, \times, <, 0, 1)$ is strictly less expressive than $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$.

As explained in Example 13.48, transitive closure logic can express the connectivity of finite graphs. Similarly, $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ 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 $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ formula $\texttt{connected}(S)$:

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

where $\texttt{lineconn}(\vec{x}, \vec{y})$ 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 $\texttt{lineconn}(D)$ (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 `connected` is a terminating formula, then this implies that `connected` expresses connectivity of semi-linear sets.

THEOREM 13.51 *The formula* `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 13.14 and Remark 13.15. So, indeed $[\mathrm{TC}_{\vec{x};\vec{y}}\, \texttt{lineconn}](\vec{s}, \vec{t})$ 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 into a finite number of convex sets (van den Dries, 1998,

Ch. 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 Fig. 13.25 for $n = 2$. QED

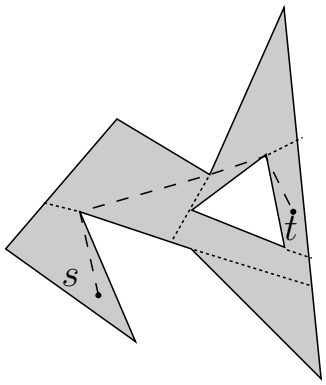

Figure 13.25. A semi-linear set decomposed in convex sets. The boundaries of the convex sets are shown in dotted lines. We have depicted a piecewise linear path (dashed line) between points s and t.

The above FO$(+, \times, <, 0, 1)$+TC-formula `connected` cannot be applied to arbitrary semi-algebraic inputs, while still guaranteeing termination. Indeed, the evaluation of `connected` on the binary relation depicted in Fig. 13.26, consisting of the points lying strictly between the parabola $y = x^2$ and the translated one $y = x^2 + 1/2$, will not terminate. 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 there exist two points in a connected component that cannot be connected by any finite piecewise linear path. Here we take as example the set defined by $(y^3 - x^2 \geq 0 \wedge x <$

Figure 13.26. 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.

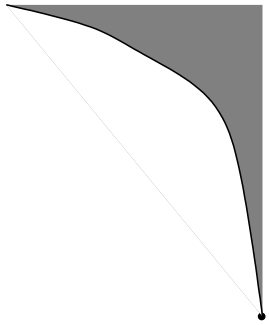

Figure 13.27. A semi-algebraic set with a cusp point.

$0 \wedge y < 1) \vee (x = 0 \wedge y = 0)$, depicted in Fig. 13.27. 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 obviously because piecewise-linear connectivity does not correspond to connectivity for arbitrary semi-algebraic sets.

Basically, Fig. 13.26 and 13.27 illustrate the only two cases where the `connected` query does not work correctly (i.e., cusp points and cusp points towards ∞).

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

THEOREM 13.52 *There exists a terminating* $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ *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 13.26), 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 $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ formula expressing connectivity.

So far, we have shown individual queries which can be expressed in $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$. Moreover, we have seen that not all formulas in $\mathrm{FO}(+, \times, <, 0, 1)+\mathrm{TC}$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ $+\mathrm{TCS}$ is built in the same way as an $\mathrm{FO}(+, \times, <, 0, 1)$ 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, σ is an $\mathrm{FO}(+, \times, <, 0, 1)$ 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

$$[\mathrm{TC}_{\vec{x};\vec{y}}\,\psi(\vec{x},\vec{y}) \mid \sigma](\vec{s},\vec{t}) \tag{13.2}$$

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

The semantics of a subformula of the above form (13.2) 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 $\mathrm{FO}(+,<,0,1)+\mathrm{TCS}$.

EXAMPLE 13.53 As an example of an $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TCS}$ formula over a two-dimensional input database S, we take

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

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 Fig. 13.23, 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 13.49). On input the graph of the function shown in Fig. 13.23, 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.

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 $\mathrm{FO}(+,\times,<,0,1)+\mathrm{KTC}$. A formula in $\mathrm{FO}(+,\times,<,0,1)+\mathrm{KTC}$ is built exactly as in $\mathrm{FO}(+,\times,<,0,1)+\mathrm{TC}$ with the exception that parameters $\vec{u}$ are allowed, i.e.,

$$\left[\mathrm{TC}_{\vec{x};\vec{y}}\psi(\vec{x},\vec{y},\vec{u})\right](\vec{s},\vec{t}) \tag{13.3}$$

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

The semantics of a subformula of the form (13.3), with $\vec{s} = (s_1, \ldots, s_k)$, 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

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

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

EXAMPLE 13.54 As an example of an $\mathrm{FO}(+,\times,<,0,1)$+KTC formula over a two-dimensional input database D, we take

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

When applied to the graph of the function, shown in Fig. 13.23, 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 $\mathrm{FO}(+,\times,<,0,1)$+KTC, the evaluation can be restricted to the computation of certain paths in the transitive closure and this gives control over the termination.

We will also consider the fragment $\mathrm{FO}(+,<,0,1)$+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 $\mathrm{FO}(+,<,0,1)$+TCS and $\mathrm{FO}(+,<,0,1)$+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 φ in $\mathrm{FO}(+,<,0,1)$+TCS (respectively in $\mathrm{FO}(+,\times,<,0,1)$+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 13.55 *Both* $\mathrm{FO}(+,<,0,1)$+TCS *and* $\mathrm{FO}(+,<,0,1)$+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 $\mathrm{FO}(+,\times,<,0,1)$+WHILE of $\mathrm{FO}(+,\times,<,0,1)$ with a while-loop.

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

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

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

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

where P is a $\mathrm{FO}(+, \times, <, 0, 1)$+WHILE program and φ an $\mathrm{FO}(+, \times, <, 0, 1)$ 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 $\mathrm{FO}(+, \times, <, 0, 1)$ 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 φ evaluates to true. One relation name R_{out} is designated as the output relation and when the $\mathrm{FO}(+, \times, <, 0, 1)$+WHILE program terminates, the current value of R_{out} is considered to be the output.

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

EXAMPLE 13.56 Let $D{=}[0, 1]$ and consider the following $\mathrm{FO}(+, \times, <, 0, 1)$+WHILE program P over the input schema $\mathcal{S} = \{S\}$, with S unary:

$R := \{(x) \mid S(x)\}$;
$Y := \emptyset$;
WHILE $R \neq \emptyset$ DO
    $Y := \{(x) \mid \varphi(x, R)\}$;
    $R := R \setminus Y$;
OD

where φ is an $\mathrm{FO}(+, \times, <, 0, 1)$ 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 is $P(D)$ is undefined. However, if we allowed infinitely long computations, then $P(D)$ would be the Cantor set which is depicted in Fig. 13.28.

Figure 13.28. The iterative construction of the Cantor set.

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

THEOREM 13.57 *The languages* $\mathrm{FO}(+,\times,<,0,1)$+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 13.55) and the fact the linearization query is expressible in $\mathrm{FO}(+,\times,<,0,1)$+TC (Theorem 13.52):

THEOREM 13.58 *The languages* $\mathrm{FO}(+,\times,<,0,1)$+TCS *and* $\mathrm{FO}(+,\times,<,0,1)$+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 $\mathrm{FO}(+,\times,<,0,1)$+TCS(respectively $\mathrm{FO}(+,\times,<,0,1)$+KTC) formula φ expressing Q on linear database, we can express Q on D in $\mathrm{FO}(+,\times,<,0,1)$+TCS(respectively $\mathrm{FO}(+,\times,<,0,1)$+KTC) by $\varphi(\widehat{D})$.

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

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

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

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

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

However, it is possible to extend $\mathrm{FO}(+,\times,<,0,1)$ directly with a connectivity operator. More generally, define a topological property Top as a collection $\{\mathcal{T}_1,\ldots,\mathcal{T}_n,\ldots\}$ 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 13.59 Of special interest is when Top expresses the property of being connected. Other examples of 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 $\mathrm{FO}(+,\times,<,0,1)+\mathbb{T}$ by extending $\mathrm{FO}(+,\times,<,0,1)$ with the following rule: if $\varphi(\vec{x},\vec{y})$ is a query then

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

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

EXAMPLE 13.60 The query "Is the intersection of regions S and T connected" can be written as $\mathsf{Conn}\vec{x} \odot S(\vec{x}) \wedge T(\vec{x})$. The query $\varphi(x) \equiv \mathsf{Conn}(y,z) \odot S(x,y,z)$ 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 13.61 *The logic* $\mathrm{FO}(+,\times,<,0,1)+\mathbb{T}$ *is closed on constraint databases.*

Proof The result follows from a simple induction on the formulas. The only case to prove is $\psi(\vec{x}) \equiv \mathsf{Top}\vec{y} \odot \varphi(\vec{x},\vec{y})$ for $\mathsf{Top} \in \mathbb{T}$. 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 Top. Therefore, the output of ψ 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 $\mathrm{FO}(+,<,0,1)+\mathbb{T}$. However, since the triviality theorem does not hold in this case, one first needs to develop an alternative decomposition theorem (see Benedikt et al., 2003 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.

Belegradek, O. V., Stolboushkin, A. P., and Taitslin, 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 Compututer 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.