



School voor Informatietechnologie

Geometric and Algorithmic Aspects of Topological Queries to Spatial Databases

Proefschrift voorgelegd tot het behalen van de graad van
doctor in de wetenschappen, richting informatica
aan de transnationale Universiteit Limburg
te verdedigen door

Floris Geerts

Promotor:
Prof. Dr. J. Van den Bussche

2001

D/2001/2451/41

Preface

First of all, I would like to thank Jan Van den Bussche for his support during the last four years. His never ending enthusiasm and advice made this dissertation possible. His intensive use of red ballpoints, learned me, hopefully, how to write scientific texts. Further, I am grateful to Bart Kuijpers. His geometrical insights were of great help. I am also grateful to Peter Revesz. The Renumbering Algorithm in Chapter 6 is the result of our joint work. I am also grateful to Bart Goethals. Apart from his enjoyable company as office-mate, his programming skills and SUN Ultra 10, made the experiments in Chapter 6 possible.

This dissertation further benefitted from pleasant discussions with, among others, Chris Giannella, Stephan Kreutzer and Leonid Libkin. I also would like to thank Masahiro Shiota for bringing Lemma I.1.3 and Lemma I.1.5 in his book [66] to my attention.

I thank the members of our research group for creating a stimulating environment.

Many thanks to all people who contributed to this dissertation in some way or another. More specifically, my thanks go to the administrative staff, colleagues, family, and friends.

Finally, I would like to thank Kristin for her support over the years.

Diepenbeek, December 2001

Contents

1	Introduction	1
2	Preliminaries	7
2.1	The Polynomial Constraint Model	7
2.2	The Linear Constraint Model	13
2.3	Generic Queries	13
2.4	Transitive Closure Logics	15
3	Expressiveness Results	19
3.1	Recursive Functions on the Natural Numbers	19
3.2	Finite Representation of \mathbf{Z} -linear Constraint Databases	21
3.3	Natural Number Representation	23
3.4	Completeness Result for \mathbf{Z} -linear Constraint Databases	26
3.5	Implications for Polynomial Constraint Databases	27
4	Geometric Properties of Semi-algebraic Sets	33
4.1	The Regular Decomposition	34
4.2	The Whitney Decomposition	36
4.3	Transversality	39
4.4	The Cone Radius	41
4.5	The Uniform Cone Radius Decomposition	46
4.6	Box Collections	49
4.7	Expressing the Box Covering Query	52
5	Linearization and Approximation	55
5.1	Construction of a Special Box Collection	55
5.1.1	Base Case: n -dimensions	56
5.1.2	Induction: from n dimensions to $n - 1$	62
5.2	The Algorithm	63
5.3	Rational Linearizations	69
5.4	The Connectivity Query	70
5.5	Volume Approximation	71

6 The Topological Invariant	77
6.1 The Maintenance of the Topological Invariant of Labeled Plane Graphs	78
6.1.1 Definitions	78
6.1.2 Data Structure and Updates	80
6.1.3 The Maintenance Algorithm	81
6.2 The Maintenance of the Topological Invariant of Arbitrary Graphs . .	87
6.2.1 Definitions	87
6.2.2 Data Structure and Updates	88
6.2.3 The Renumbering Algorithm	90
6.2.4 The Topology Tree Algorithm	94
6.2.5 Experimental Results	100
Bibliography	103
Index	109
Samenvatting	111

1

Introduction

Spatial database systems [1, 9, 19, 34, 35, 63] are concerned with the representation and manipulation of data that has a geometric or topological interpretation. Conceptually, spatial databases store geometric figures, which are possibly infinite sets of points in a real space \mathbf{R}^n . The framework of constraint databases [51], introduced by Kanellakis, Kuper, and Revesz [39], provides an elegant and powerful model for spatial databases. In the setting of the constraint model, a geometric figure is finitely represented as a Boolean combination of polynomial equalities and inequalities over the real numbers. Such figures are known as semi-algebraic sets. The special case of figures definable by linear polynomials are known as semi-linear sets.

The relational calculus (first-order logic), expanded with polynomial equalities and inequalities and evaluated over the semi-algebraic sets (viewed as relations over the reals) stored in the database, serves as a basic spatial query language, and is denoted by FO+POLY. The special case of queries expressed using linear equalities and inequalities is denoted by FO+LIN. Several authors have argued that the restriction to linear polynomial constraints provides a sufficiently general framework for spatial database applications [31, 73, 74]. Indeed, in geographic information systems (GIS), which form one of the main application areas of spatial databases, linear representations are used to model spatial objects [51, Chapter 9]. Existing implementations of the constraint model, for instance, the work on the system DEDALE [29, 30, 31], are also restricted to linear polynomial constraints. The evaluation of queries expressed in FO+LIN is conceptually easy and can be computed by numerous efficient algorithms for geometric operations on linear figures [60]. The computational complexity of evaluating an FO+LIN query on linear constraint databases (NC^1) is also slightly lower than that of evaluating an FO+POLY query on polynomial constraint databases (NC) [3, 13, 32, 62, 69].

Since the expressive power of the basic query languages FO+POLY and FO+LIN is rather limited [51, Chapter 5 and 6], it makes sense to consider more powerful extensions.

Extensions with Recursion Various extensions with recursion have been introduced and studied. Grumbach and Kuper [28] defined syntactic variants of DATALOG with linear constraints which capture exactly the queries on linear constraint databases in the plane, which have PTIME and PSPACE data complexity. Kreutzer [45] defines several recursive languages capturing PTIME and PSPACE on a restricted class of linear constraint databases. Termination properties of DATALOG with polynomial constraints are investigated by Kuijpers, Smits, and Van den Bussche [48, 50].

In this dissertation, we study the expressive power of FO+POLY (and FO+LIN) extended with the transitive closure operator TC. Transitive closure is a simple form of recursion and we only apply it in a simple way, i.e., we do not apply TC to formulae with extra free variables (parameters), as is allowed in the standard definition of transitive closure logic [17].

In Chapter 3, we obtain the following result: when we extend the TC operator with explicit stop conditions, which we denote by TCS, the language FO+LIN+TCS is computationally complete on the class of \mathbf{Z} -linear databases, i.e., databases which can be defined by linear polynomials with integer coefficients. This means that for every partial computable query Q , there is a formula φ such that for every \mathbf{Z} -linear database D , the evaluation of φ on D terminates if and only if $Q(D)$ is defined and results in $Q(D)$. It remains an open problem whether FO+LIN+TC (without explicit stop conditions) is also computationally complete in this sense.

Whether FO+POLY+TCS is computationally complete on polynomial constraint databases, also remains an open problem. Recently, Kreutzer [44] defined a different extension of FO+LIN with a transitive closure operator and also obtained the computational completeness of this language. Koiran, Cosnard, and Garzon [41] investigated similar issues in the context of piecewise linear dynamical systems. Other computationally complete languages for linear constraint queries are studied by Gyssens, Van den Bussche, and Van Gucht [36].

Linearization Although it is not clear whether FO+POLY+TCS is computationally complete on polynomial constraint databases, we obtain an expressivity result of FO+POLY+TCS for a restricted class of queries. This class of queries, important for many spatial database applications, consists of those queries involving only topological properties of the database, which therefore is called the class of topological queries. For instance, the query which asks whether a database is open or connected, is a topological query. The query which asks whether the distance between two points in the database is 5 cm, is an example of a query which is not topological.

The research on topological properties of spatial databases is focused on two topics, the first being topological query languages. For instance, a query language based on topological information of a set of regions is studied by Egenhofer [18]. Region-based query languages date back to Clark [11], and have been used in reasoning about

spatial knowledge in Artificial Intelligence [12, 59]. We also refer to the region-based languages of Papadimitriou, Suciu, and Vianu [57]. The expressivity of FO+POLY, with respect to topological queries, is investigated by Kuijpers, Paredaens, and Van den Bussche [49], Segoufin and Vianu [64], and Grohe and Segoufin [27].

In Chapters 4 and 5, we prove that there is a formula in FO+POLY+TC that expresses linearization. When evaluated on an arbitrary semi-algebraic set A , the formula results in a semi-linear set \widehat{A} , topologically equivalent (i.e., homeomorphic) to A . Moreover, \widehat{A} is \mathbf{Z} -linear.

The linearization formula always terminates, in the sense that on any input A , every application of the TC operator in the formula converges after a finite number of stages. As a consequence, the language FO+POLY+TCS is computationally complete on the class of polynomial constraint databases as far as all boolean topological queries are concerned.

Benedikt, Grohe, Libkin, and Segoufin [5] proved that extending FO+POLY with a topological property results in a closed query language, i.e., the result of a query in this language is still a polynomial constraint database. Adding all topological properties to FO+POLY clearly results in a language which is complete with respect to Boolean topological queries. However, we believe it is more elegant to extend FO+POLY with a single new feature, like a transitive closure operator, instead of extending FO+POLY with an uncountable number of new features (there are uncountable many topological properties).

The FO+POLY+TC linearization formula described above also has other consequences. For instance, in Chapter 5, we prove that the connectivity query on polynomial constraint databases is expressible by an always terminating formula in plain FO+POLY+TC. Since DATALOG with polynomial constraints contains FO+POLY+TC, the connectivity query is also expressible in DATALOG with polynomial constraints. This answers the question, raised by Kuijpers and Smits [50], which was already partially solved [23, 24], affirmatively.

The linearization formula can be sharpened so as to result in a set \widehat{A} that is arbitrarily close to the input set A , on condition that A is bounded. There is an always terminating formula in FO+POLY+TC that evaluates on a given bounded semi-algebraic set A to a number that is arbitrarily close to the volume of A . Other techniques for approximating the volume in extensions of FO+POLY can be found in [7].

The Topological Invariant The second topic in the research on topological properties of spatial databases is the representation of topological information. It is known that for polynomial constraint databases in the plane, there exists a finite combinatorial structure which is called the topological invariant and captures exactly the topological information in the database. Intuitively, the invariant is obtained as follows: first, the “cells” need to be identified, i.e., the points, lines, and faces that are topologically significant. This can be done by using the well-known cell decomposition of polynomial constraint databases [8, 37, 42, 62]. Next, the adjacency relationships between the cells, i.e., endpoints of lines, boundaries of faces and so forth, need to be

identified, as well as the external face and the circular list of lines and faces adjacent to each point.

There exists a unique cell decomposition with a minimal number of cells. The finite structure describing this cell decomposition is called the topological invariant and is known to be complete [47, 57]. This means that two planar polynomial constraint databases which have isomorphic topological invariants, are indistinguishable from a topological point of view. In the theory of spatial query languages, the topological invariant is used to obtain expressiveness results [64]. More specifically, topological queries are translated to its topological invariant.

Of course, if we want to answer queries using the topological invariant instead of the original polynomial constraint database, we are faced with the online maintenance of the topological invariant under updates to the original polynomial constraint database. The class of online algorithms which react on updates are called dynamic algorithms and can be divided into fully dynamic algorithms, which react correctly on all possible updates, and partially dynamic algorithms, which react correctly on only one type of update [2, 20].

In Chapter 6, we consider the problem of maintaining the topological invariant of a polynomial constraint database. We note that the cell decomposition described above provides another way to model a planar spatial database, i.e., as a planar subdivision consisting of points, lines, and areas, which is the approach taken in many geographical information systems [70]. The common data structure used to represent planar subdivisions is the plane graph structure [43], perhaps better known as the doubly connected edge list [60, 15].

We give a fully dynamic algorithm which maintains the topological invariant of a labeled planar graph, representing a polynomial constraint database. The algorithm is built on top of the doubly connected edge list, and has complexity $O(\ell)$ per update, with ℓ the current size of the invariant.

We then focus on general graphs, i.e., we do not make any assumptions on the graph such as planarity and the like, and provide two partially dynamic algorithms for maintaining the topological invariant of a graph. The algorithms are partially dynamic because we only admit edge insertions. Both algorithms have complexity $O(\log \ell)$ per update, with ℓ the number of edges. The topological invariant of a graph is obtained by eliminating all “regular” vertices, which are the vertices that are part of chain; in graph theoretic terms they are the vertices of degree two.

Regular vertices often occur abundantly. For example, in a road network, each bend in the road is represented by a vertex, which will be regular. The same applies more generally to all networks represented on top of a discrete raster [80], where a curved line is approximated by many straight line segments between raster points, which will then be regular.

Since both algorithms have the same complexity, we have performed an empirical study on their relative performance. More precisely, we implemented both algorithms and performed experiments on input graphs subject to edge insertions only.

Overview The following chapters are organized as follows. Chapter 2 provides the necessary background on constraint databases and query languages. It gives the definition of topological queries and introduces transitive closure logics. Chapter 3 shows that $\text{FO}+\text{LIN}+\text{TCS}$ is computationally complete on \mathbf{Z} -linear constraint databases, and that $\text{FO}+\text{POLY}+\text{TCS}$ is computationally complete on polynomial constraint databases with respect to Boolean topological queries.

Chapter 4 looks at the geometric properties of polynomial constraint databases and shows that many of these properties are expressible by first-order means. More specifically, we define the local cone structure of polynomial constraint databases for boxes, and prove that this is a first-order expressible property. We conclude this chapter by defining the uniform cone radius decomposition and the notion of a box collection, which we will use in Chapter 5. In that chapter, we construct a special box collection and show how it can be used to construct a linearization of a polynomial constraint database. We then show that this construction is expressible in $\text{FO}+\text{POLY}+\text{TC}$. As a consequence, we show that the connectivity query is expressible in $\text{FO}+\text{POLY}+\text{TC}$. After a minor adaptation of the linearization, we show how it can be used to approximate the volume of a polynomial constraint database. Finally, Chapter 6 deals with the online maintenance of the topological invariant.

2

Preliminaries

In this chapter, we formally define the polynomial constraint database model as an extension of the classical relational database model, and the linear constraint database model as a restriction of the polynomial constraint model. We introduce the concept of query in these models and define the basic constraint query language for both models.

We discuss the notion of genericity in the context of constraint databases and look closer to queries which are generic with respect to topological transformations, i.e., homeomorphisms.

Next, the basic constraint query languages are augmented with a very simple recursion mechanism, namely a transitive closure operator, which is denoted by TC. Transitive closure is a very simple form of recursion, moreover, we do not use applications of TC to formulae with extra free variables (parameters), as is allowed in the standard definition of transitive closure. We also extend the TC operator with explicit stop conditions, which results in a very expressive query language, as will be shown in the next chapter.

2.1 The Polynomial Constraint Model

In the polynomial constraint model, geometric data which can be defined in first-order logic over the real numbers with addition and multiplication is considered. Geometric data can be represented with first-order logic formulae as follows. Assume an infinite set of variables which range over the real numbers. We call these variables real variables and denote them with x_1, x_2, \dots . Define a polynomial constraint term as a polynomial in real variables with algebraic coefficients. An atomic polynomial

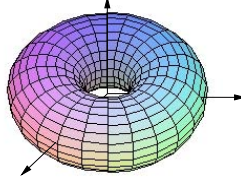


Figure 2.1: Example of a semi-algebraic figure.

constraint formula is built from a polynomial constraint term using binary comparison relations, i.e., $t \theta 0$ where t is a polynomial constraint term and $\theta \in \{=, <, >, \leq, \geq, \neq\}$. A *polynomial constraint formula* is defined inductively as follows:

- every atomic polynomial constraint formula is a polynomial constraint formula;
- if φ and ψ are polynomial constraint formulae, then $\varphi \wedge \psi$ and $\neg\varphi$ are polynomial constraint formulae; and
- if x is a real variable and φ is a polynomial constraint formula in which x occurs free, then $(\exists x)\varphi(x)$ is a polynomial constraint formula.

Every polynomial constraint formula φ with n free real variables x_1, \dots, x_n defines a geometric figure

$$\{(x_1, \dots, x_n) \in \mathbf{R}^n \mid \varphi(x_1, \dots, x_n)\}$$

in the n -dimensional real space \mathbf{R}^n .

Example 2.1.1. The polynomial constraint formula

$$\begin{aligned} x^4 + y^4 + z^4 + 2x^2y^2 + 2x^2z^2 + 2y^2z^2 - 2(r_0^2 + r_1^2)x^2 \\ + 2(r_0^2 - r_1^2)y^2 - 2(r_0^2 + r_1^2)z^2 + (r_0^2 - r_1^2)^2 = 0 \end{aligned}$$

defines the torus shown in Figure 2.1 for $r_0 = 2$ and $r_1 = 1/2$ (r_0 is the major radius of the torus, r_1 is the minor radius). \square

We now define the class of figures which correspond to the geometric figures described by polynomial constraint formulae. A basic semi-algebraic set in (Euclidean space) \mathbf{R}^n is a point set

$$\{(x_1, \dots, x_n) \in \mathbf{R}^n \mid p(x_1, \dots, x_n) > 0\},$$

where $p(x_1, \dots, x_n)$ is a polynomial with algebraic coefficients in the real variables x_1, \dots, x_n . A *semi-algebraic set* in \mathbf{R}^n is inductively defined as follows:

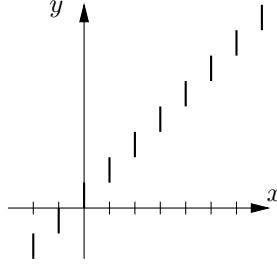


Figure 2.2: An example of a set which is not semi-algebraic.

- A basic semi-algebraic set in \mathbf{R}^n is a semi-algebraic set in \mathbf{R}^n ; and
- if A and B are two semi-algebraic sets in \mathbf{R}^n , then $A \cap B$ and $\mathbf{R}^n - A$ are semi-algebraic sets in \mathbf{R}^n .

Of course, not every set is semi-algebraic.

Example 2.1.2. The geometric figure defined as

$$\{(x_1, x_2) \in \mathbf{R}^2 \mid x_1 \in \mathbf{Z} \wedge x_1 \leq x_2 \leq x_1 + 1\}$$

is not semi-algebraic (see Figure 2.2). \square

It is clear that semi-algebraic sets in \mathbf{R}^n are exactly the sets in \mathbf{R}^n described by a quantifier free polynomial constraint formula.

We refer to Bochnak, Coste and Roy [8], and Benedetti and Rissler [4] for an exposition of real algebraic geometry and properties of semi-algebraic sets.

Tarski-Seidenberg Theorem ([65, 69]). *Let A be a semi-algebraic set in \mathbf{R}^{n+1} , and let $\Pi : \mathbf{R}^{n+1} \rightarrow \mathbf{R}^n$ be the projection on the space of the first n coordinates. Then $\Pi(A)$ is a semi-algebraic set in \mathbf{R}^n .* \square

The logical equivalent of this theorem is that first-order logic over the real numbers with addition and multiplication has *quantifier elimination*. This means that every polynomial constraint formula is equivalent to a quantifier free polynomial constraint formula.

The existence of quantifier elimination for polynomial constraint formulae follows directly from the Tarski-Seidenberg Theorem. Indeed, let $\varphi(x_1, \dots, x_n, x_{n+1})$ be a quantifier free polynomial constraint formula, and let $A = \{(x_1, \dots, x_{n+1}) \in \mathbf{R}^{n+1} \mid \varphi(x_1, \dots, x_{n+1})\}$. Then $A' = \{(x_1, \dots, x_n) \in \mathbf{R}^n \mid (\exists x_{n+1})\varphi(x_1, \dots, x_{n+1})\}$ is equal to $\Pi(A)$, with Π the natural projection $\Pi : \mathbf{R}^{n+1} \rightarrow \mathbf{R}^n$. By the Tarski-Seidenberg Theorem, the set A' is also semi-algebraic, and hence can be defined with quantifier free polynomial constraint formulae only.

Example 2.1.3. Let $\varphi(a, b, c, d)$ be the polynomial constraint formula

$$(\exists x)(\exists y)(\exists u)(\exists v)(xa + yc = 1 \wedge xb + yd = 0 \wedge ua + vc = 0 \wedge ub + vd = 1).$$

The formula $\varphi(a, b, c, d)$ asserts that the matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is invertible. It is well-known that $\varphi(a, b, c, d)$ is equivalent to the quantifier free polynomial constraint formula $ad - bc \neq 0$. \square

Later in this chapter we will see the application of quantifier elimination in the context of query evaluation.

We proceed with introducing the polynomial constraint database model. In the relational database model, a relation consists of columns that store values of some alpha-numerical data type. The polynomial constraint database model extends this model by adding an extra geometric column that store semi-algebraic sets. In contrast with the alpha-numerical data columns, there is a sharp distinction between what is stored in a geometric column (quantifier free polynomial constraint formulae) and the interpretation of the stored data (geometric figures in Euclidean space).

Formally, we define a polynomial constraint database *scheme*, \mathcal{S} , as a finite set of relation names. With each relation name, R , a type is associated. A type is a pair of natural numbers, $[m, n]$, where m denotes the number of alpha-numerical columns, and n denotes the dimension of the single geometric column of R . A polynomial constraint database schema has type $[m_1, n_1; \dots; m_k, n_k]$ if the schema consists of relation names, R_1, \dots, R_k , of type $[m_1, n_1], \dots, [m_k, n_k]$ respectively. A syntactic polynomial constraint relation of type $[m, n]$ is a finite set of tuples of the form

$$(v_1, \dots, v_m; \varphi(x_1, \dots, x_n))$$

with v_1, \dots, v_m being alpha-numerical values of some domain U , and $\varphi(x_1, \dots, x_n)$ a quantifier free polynomial constraint formula with n free real variables. A *syntactic polynomial constraint database instance*, or shortly, a polynomial constraint database, is a mapping \mathcal{I} , assigning to each relation name R of a schema \mathcal{S} , a syntactic polynomial constraint relation $\mathcal{I}(R)$ of the same type.

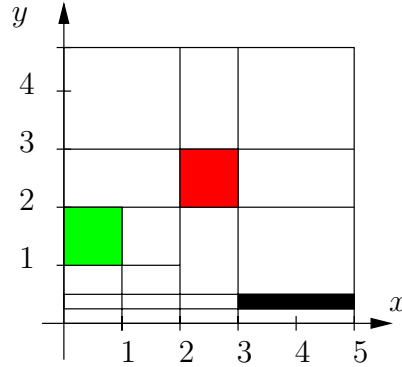
Given a syntactic polynomial constraint relation R , the semantic polynomial constraint relation, $I(R)$, is defined as

$$\bigcup_{t \in R} \{(t.v_1, \dots, t.v_m)\} \times \{(u_1, \dots, u_n) \in \mathbf{R}^n \mid t.\varphi(u_1, \dots, u_n)\}.$$

Given a syntactic polynomial constraint database instance \mathcal{I} over a polynomial constraint database scheme \mathcal{S} , the *semantic polynomial constraint database instance* is the mapping I , assigning to each relation name R in \mathcal{S} the semantic polynomial constraint relation $I(\mathcal{I}(R))$.

Example 2.1.4. In this example a piece of a polynomial constraint database containing information on paintings in a museum of modern art is shown. The schema has type $[4, 0; 2, 2]$. The first relation consists entirely of alpha-numerical data containing the name of the artist, the name of the work, the year of creation, and an

ID number of the piece of art. The second relation consists of an alpha-numerical part, containing an ID number and a name of a color. The geometric part contains the polynomial constraint representation of parts of the painting, which are colored according to the color in the second column.



Artist	Work	Year	ID
Piet Mondriaan	Composition with red, green and black	2001	1

ID	Color	Geometry
1	Red	$2 < x < 3 \wedge 2 < y < 3$
1	Green	$0 < x < 1 \wedge 1 < y < 2$
1	Black	$((y = 0.25 \vee y = 0.5 \vee y = 2 \vee y = 3) \wedge 0 \leq x \leq 5) \vee ((x = 2 \vee x = 3) \wedge 0 \leq y \leq 5) \vee (3 \leq x \leq 5 \wedge 0.25 \leq y \leq 0.5)$

□

In standard relational databases, a query is a mapping associating to each database an answer relation. In the polynomial constraint model, the picture is somewhat more complicated. Given an input scheme \mathcal{S}_{in} and an output scheme \mathcal{S}_{out} , a query is a mapping of the polynomial constraint database instances of \mathcal{S}_{in} to the polynomial constraint relation instances of \mathcal{S}_{out} , at both the syntactic and the semantic level.

We associate with every query a type

$$[m_1, n_1; \dots; m_k, n_k] \rightarrow [m, n]$$

with $[m_1, n_1; \dots; m_k, n_k]$ the type of the input database scheme and $[m, n]$ the type of the output database scheme.

By adding to the language of polynomial constraint formulae the following:

- a totally ordered infinite set of variables, called value variables, disjoint from the set of real variables, which range over the alpha-numerical domain U ;
- atomic formulae of the form $v_1 = v_2$, with v_1 and v_2 value variables;
- atomic formulae of the form $R(v_1, \dots, v_m; x_1, \dots, x_n)$, with R a relation name of type $[m, n]$, v_1, \dots, v_m value variables, and x_1, \dots, x_n real variables; and
- existential quantification of value variables,

we obtain a query language, which is known as the *polynomial constraint calculus*, and is denoted by FO+POLY. We say that φ is an FO+POLY formula over schema \mathcal{S} , if the relation names occurring as atomic subformulae in φ are in \mathcal{S} .

A query of type $[m_1, n_1; \dots; m_k, n_k] \rightarrow [m, n]$ is expressible in FO+POLY if there exists an FO+POLY formula φ with m free value variables and n free real variables such that at the semantic level, for every input database instance of type $[m_1, n_1; \dots; m_k, n_k]$ the output database equals

$$\{(v_1, \dots, v_m; x_1, \dots, x_n) \mid \varphi(v_1, \dots, v_m, x_1, \dots, x_n)\}.$$

Queries of type $[m_1, n_1; \dots; m_k, n_k] \rightarrow [0, 0]$ are called *Boolean queries*, because the sets $\{()\}$ and $\{\}$ can be seen as encoding, the truth values *true* and *false* respectively.

Notational Convention. In all queries below, the input database schema \mathcal{S} consists of relation names S_1, \dots, S_k of purely geometric type $[0, n_1; \dots; 0, n_k]$. We then say that S_i has *arity* n_i , for $i = 1, \dots, k$. If D is a polynomial constraint database over schema \mathcal{S} , then we denote the semantic polynomial database instance $I(D(S_i))$ by S_i^D , for $i = 1, \dots, k$. If φ is a formula in FO+POLY over \mathcal{S} , then we denote the output semantic polynomial database instance by $\varphi(S_i)$, for $i = 1, \dots, k$. We will represent n -tuples of variables x_1, \dots, x_n , with \vec{x} .

Example 2.1.5. Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. The query which asks whether a database is bounded, is expressible in FO+POLY. Indeed, define the formula over \mathcal{S}

$$\text{bounded} \equiv \exists M \forall \vec{x} (S(\vec{x}) \rightarrow x_1^2 + x_2^2 + \dots + x_n^2 < M).$$

Then, for any database D over \mathcal{S} , $\text{bounded}(D)$ is *true* if and only if $S^D \subseteq \mathbf{R}^n$ is bounded. \square

FO+POLY queries can be effectively evaluated as follows. Let $\varphi(x_1, \dots, x_k)$ be an FO+POLY formula over schema \mathcal{S} , and let D be a database over \mathcal{S} . For every $S \in \mathcal{S}$, we represent the set S^D by some quantifier-free polynomial constraint formula $\psi_S(y_1, \dots, y_k)$, where k is the arity of S that defines S^D in the sense that $S^D = \{(a_1, \dots, a_n) \in \mathbf{R}^n \mid \psi_S(a_1, \dots, a_n)\}$. Now replace in φ every subformula of the form $S(z_1, \dots, z_n)$ by $\psi_S(z_1, \dots, z_n)$. Doing these replacements for every $S \in \mathcal{S}$ we obtain a polynomial constraint formula which we denote by φ^D , and which defines $\varphi(D)$ in the sense that $\varphi(D) = \{(a_1, \dots, a_k) \in \mathbf{R}^k \mid \psi_S(a_1, \dots, a_k)\}$.

By the Tarski-Seidenberg Theorem, we can rewrite φ^D in a quantifier-free form from which we see that $\varphi(D)$ is a semi-algebraic set. This is called the closure principle.

2.2 The Linear Constraint Model

When only linear polynomials are considered in the polynomial constraint model, one obtains the so-called linear constraint database model.

We distinguish between two different linear constraint models, depending on the choice of coefficients in the linear polynomials.

We define a **Z**-linear constraint term as a polynomial in real variables with integer coefficients. An atomic **Z**-linear constraint formula is built from a **Z**-linear constraint term using binary comparison relation, i.e., $t\theta 0$, where t is a **Z**-linear constraint term and $\theta \in \{=, <, >, \leq, \geq, \neq\}$. Polynomial constraint formulae built from atomic **Z**-linear constraint formulae are called ***Z**-linear constraint formulae*. The **Z**-linear constraint formulae are equivalent to first-order formulae over the real numbers with addition.

A **Z**-linear constraint formula $\varphi(x_1, \dots, x_n)$ with n free real variables x_1, \dots, x_n defines a geometric figure $\{(x_1, \dots, x_n) \mid \varphi(x_1, \dots, x_n)\}$ in \mathbf{R}^n by allowing the real variables to range over the real numbers.

Using algebraic computational techniques for the elimination of variables in sets of linear equalities and inequalities, one obtains an exact correspondence between these figures defined by a **Z**-linear constraint formula and the so-called **Z**-linear sets.

Polynomial constraint databases containing only **Z**-linear sets as geometric data are called ***Z**-linear constraint databases*. The syntactic and semantic relations of a **Z**-linear constraint database are called, syntactic and semantic **Z**-linear constraint relations respectively.

When we allow real algebraic coefficients in the linear polynomials, we talk about **A**-linear constraint terms, atomic **A**-linear constraint formulae, **A**-linear sets, **A**-linear constraint relations and **A**-linear constraint databases.

The restrictions of the query language FO+POLY to the context of **Z**-linear constraint database, and to the context of **A**-linear databases, are denoted by FO+**Z**-LIN, and FO+**A**-LIN respectively.

Formulae in FO+**Z**-LIN (FO+**A**-LIN) also admit quantifier elimination, so if D is a **Z**-linear (**A**-linear) database, and φ is in FO+**Z**-LIN (FO+**A**-LIN), then also $\varphi(D)$ is **Z**-linear (**A**-linear). Hence, there is also a closure principle for FO+**Z**-LIN and FO+**A**-LIN, provided we work with **Z**-linear (**A**-linear) database only.

We shall talk about linear constraint terms, atomic linear constraint formulae, linear constraint formulae, semi-linear sets, linear constraint relations, linear constraint databases, linear queries, and FO+LIN whenever the distinction between integer or algebraic coefficients is irrelevant.

2.3 Generic Queries

In the context of the relational database model, Chandra and Harel [10] investigated which queries are “reasonable”. They characterized this class of queries by means of the concept of genericity. Informally, this means that output of a generic query is independent of the internal representation of the data and only depends on the

logical structure of the databases. Paredaens, Van den Bussche and Van Gucht [58] have shown that for constraint databases, the definition of genericity depends on the particular kind of geometry in which the spatial information is to be interpreted.

Let φ be a query of type $[0, n; \dots; 0, n] \rightarrow [0, m]$ and let \mathcal{G} be a group of transformations of \mathbf{R}^n . The query Q is \mathcal{G} -generic if, for every transformation $g \in \mathcal{G}$ and any two input database instances D_1 and D_2 of type $[0, n; \dots; 0, n]$,

$$g(D_1) = D_2 \rightarrow g(Q(D_1)) = Q(D_2), \quad (2.3.1)$$

where the transformation g is assumed to canonically extend from \mathbf{R}^n to \mathbf{R}^{kn} , for any value of $k \in \mathbf{N}$.

Example 2.3.1. Let \mathcal{A} be the group of affinities. Then, the query which asks whether the database lies on a straight line, is \mathcal{A} -generic since affinities preserve collinearity. \square

We are interested in queries which are generic for the group of homeomorphisms, which we denote by \mathcal{H} . A *homeomorphism* h of \mathbf{R}^n is a bijective mapping from \mathbf{R}^n to \mathbf{R}^n such that both h and its inverse h^{-1} are continuous. Two subsets X and Y of \mathbf{R}^n are *topologically equivalent* if there exists a homeomorphism h of \mathbf{R}^n such that $h(X) = Y$.

Example 2.3.2. A square and a circle are topologically equivalent. Indeed, let $S^1 = \{(x, y) \in \mathbf{R}^2 \mid x^2 + y^2 = 1\}$ be the unit circle in \mathbf{R}^2 , and let $Sq = \{(x, y) \in \mathbf{R}^2 \mid \max\{|x|, |y|\} = 1\}$ be the square of size 2 centered at the origin. Consider the mapping

$$h : \mathbf{R}^2 \rightarrow \mathbf{R}^2 : (x, y) \mapsto \begin{cases} \frac{x}{\sqrt{x^2+y^2}}(x, y) & \text{if } x \geq 0 \wedge x \geq y \\ \frac{-x}{\sqrt{x^2+y^2}}(x, y) & \text{if } x < 0 \wedge x \geq y \\ \frac{y}{\sqrt{x^2+y^2}}(x, y) & \text{if } y \geq 0 \wedge y \geq x \\ \frac{-y}{\sqrt{x^2+y^2}}(x, y) & \text{if } y < 0 \wedge y \geq x \\ (0, 0) & \text{if } (x, y) = (0, 0). \end{cases}$$

This mapping h projects each point on Sq radially inward to the sphere S^1 (see Figure 2.3). It is easy to show that h is a homeomorphism of \mathbf{R}^2 and $h(Sq) = S^1$. \square

Following the definition of genericity, a query is *topological* if it is \mathcal{H} -generic.

Example 2.3.3. The query of Example 2.3.1, which asks whether a polynomial constraint database lies on a straight line is not topological. The *connectivity query*, which asks whether a polynomial constraint database is connected, is an example of a topological query. The query expressed by the formula

$$\exists \varepsilon (\varepsilon \neq \vec{0} \wedge \forall \vec{y} (\vec{x} - \varepsilon < \vec{y} < \vec{x} + \varepsilon \rightarrow S(\vec{y})))$$

returns the interior of the semi-algebraic set S^D when evaluated on any polynomial constraint database D . This query is a topological query. \square

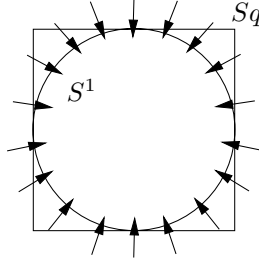


Figure 2.3: A square and a circle are topologically equivalent.

2.4 Transitive Closure Logics

Many interesting spatial database queries are not expressible in the first-order query languages FO+POLY and FO+LIN, for example, the query that asks whether a given database is topologically connected. Therefore, it makes sense to consider extensions of FO+POLY (or FO+LIN) with recursion to obtain more powerful query languages. We study one of the most simple recursion constructs in this context, i.e., the transitive closure operator TC. An immediate observation is that TC cannot be added just like that with its standard mathematical semantics, without losing the important closure principle. For example, the transitive closure of the semi-algebraic set $\{(x, y) \in \mathbf{R}^2 \mid y = 2x\}$ equals $\{(x, y) \in \mathbf{R}^2 \mid \exists i \in \mathbf{N} : y = 2^i x\}$, which is not semi-algebraic.

Therefore, we look at the TC operator quite naturally as a programming construct with a purely operational semantics. For example, we will look at the transitive closure example just mentioned simply as a non-terminating computation. Almost all programming languages allow the expression of non-terminating computations, and it is part of the programmer's job to avoid writing such programs.

A formula in FO+POLY+TC is a formula built in the same way as an FO+POLY formula, but with the following extra formation rule: if $\psi(\vec{x}, \vec{y})$ is a formula with \vec{x}, \vec{y} k -tuples of variables, and \vec{s}, \vec{t} are k -tuples of variables, then

$$[\text{TC}_{\vec{x}, \vec{y}} \psi](\vec{s}, \vec{t}) \quad (2.4.1)$$

is also a formula which has as free variables those in \vec{s} and \vec{t} . Since the only free variables in $\psi(\vec{x}, \vec{y})$ are those in \vec{x} and \vec{y} , we do not allow parameters in applications of the TC operator, as is allowed in finite model theory. With parameters, it is not so clear how to preserve the simple and elegant operational semantics we define next.

The semantics of a subformula of the above form (2.4.1) evaluated on a database D is defined in the following operational manner:

1. Evaluate, recursively, $\psi(D)$.
2. Start computing the following iterative sequence of $2k$ -ary relations:

$$\begin{aligned} X_0 &:= \psi(D) \\ X_{i+1} &:= X_i \cup \{(\vec{x}, \vec{y}) \in \mathbf{R}^{2k} \mid \exists \vec{z} X_i(\vec{x}, \vec{z}) \wedge X_i(\vec{z}, \vec{y})\}. \end{aligned}$$

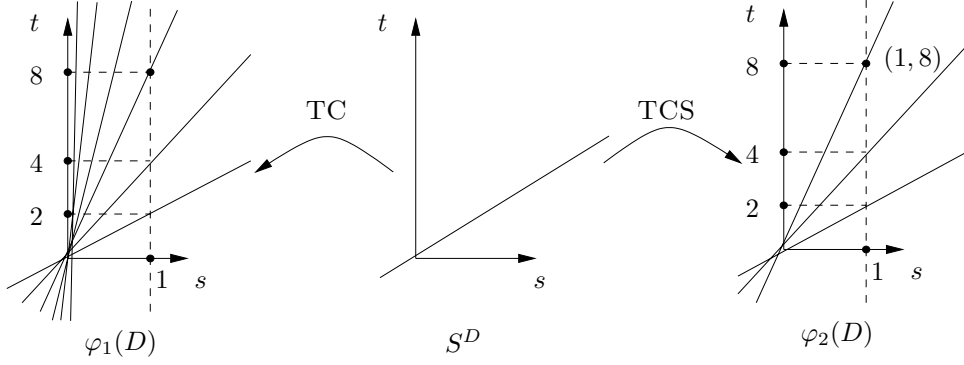


Figure 2.4: Illustration of the difference between transitive closure without stop condition (left) and with stop condition (right).

3. Stop as soon as an i has been found such that $X_i = X_{i+1}$.
4. Evaluate the formula $X(\vec{s}, \vec{t})$ where relation name X has the relation X_i as value.

Since every step in the above algorithm, including the test for $X_i = X_{i+1}$, is expressible in FO+POLY, every step is effective and the only reason why the evaluation may not be effective is that the computation does not terminate. In that case the evaluation of the formula 2.4.1 (and any other formula in which it occurs as subformula) is undefined. The language FO+LIN+TC consists of all FO+POLY+TC programs that do not use multiplication.

Example 2.4.1. Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. Consider the following FO+POLY+TC formula over \mathcal{S} :

$$\mathbf{connected} \equiv \forall \vec{x} \forall \vec{y} S(\vec{x}) \wedge S(\vec{y}) \rightarrow [\mathbf{TC}_{\vec{r}, \vec{s}} \mathbf{lineconn}](\vec{x}, \vec{y})$$

where $\mathbf{lineconn}(\vec{r}, \vec{s})$ is the formula

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

In Chapter 5, we will prove that the TC-subformula in **connected** terminates on all linear constraint databases over \mathcal{S} , and asks whether the linear constraint database is connected. \square

We will sometimes need to be able to specify an explicit termination condition on transitive closure computations. Thereto we introduce the language FO+POLY+TCS.

Formulae in FO+POLY+TCS are again built in the same way as in FO+POLY but with the following extra formation rule: if $\psi(\vec{x}, \vec{y})$ is a formula with \vec{x}, \vec{y} k -tuples of variables; σ is an FO+POLY sentence (formula without free variables) over the schema \mathcal{S} expanded with a special $2k$ -ary relation name X ; and \vec{s}, \vec{t} k -tuples of variables, then

$$[\mathbf{TC}_{\vec{x}, \vec{y}} \psi \mid \sigma](\vec{s}, \vec{t}) \quad (2.4.2)$$

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 (2.4.2) 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 in case an i is found such that $(D, X_i) \models \sigma$, whatever case occurs first.

Example 2.4.2. Let $\mathcal{S} = \{S\}$, with S an n -ary relation in name. Consider the FO+POLY+TCS formula

$$\varphi_1(s, t) \equiv [\text{TC}_{x;y}S](s, t) \tag{2.4.3}$$

and the formula

$$\varphi_2(s, t) \equiv [\text{TC}_{x;y}S \mid X(1, 8)](s, t). \tag{2.4.4}$$

On the polynomial constraint database D over S , where $S^D = \{(x, y) \in \mathbf{R}^2 \mid y = 2x\}$, the evaluation of formula (2.4.3) does not terminate, but formula (2.4.4) evaluates in 3 iterations to $\{(s, t) \in \mathbf{R}^2 \mid t = 2s \vee t = 4s \vee t = 6s \vee t = 8s\}$ (see Figure 2.4). \square

The language FO+LIN+TCS consists of all FO+POLY+TCS programs that do not use multiplication.

3

Expressiveness Results

In this chapter, we show a general result on the expressive power of $\text{FO}+\text{LIN}+\text{TCS}$. More specifically, we prove that $\text{FO}+\text{LIN}+\text{TCS}$ is computationally complete on \mathbf{Z} -linear constraint databases (Theorem 3.4.1). The proof consists of three steps. In the first step, we show that any computable function on the natural numbers can be simulated by an $\text{FO}+\text{LIN}+\text{TCS}$ expression (Lemma 3.1.1). In the second step, we show that there exists an encoding of \mathbf{Z} -linear constraint databases by finite sets of rational numbers, and show that both the encoding and the corresponding decoding are in $\text{FO}+\text{LIN}+\text{TCS}$ (Lemma 3.2.1 and Lemma 3.3.1). In the third step, we show that there exists an encoding of finite sets of rational numbers by natural numbers, and show that both the encoding and the corresponding decoding are in $\text{FO}+\text{LIN}+\text{TCS}$. This implies that $\text{FO}+\text{LIN}+\text{TCS}$ is computationally complete on \mathbf{Z} -linear constraint databases.

For polynomial constraint databases we show that $\text{FO}+\text{POLY}+\text{TCS}$ is computationally complete for Boolean topological queries. This follows from the completeness on \mathbf{Z} -linear constraint databases and the existence of an $\text{FO}+\text{POLY}+\text{TC}$ query that, given any polynomial constraint database as input, returns a \mathbf{Z} -linear constraint database which is topological equivalent to the input. In this chapter we show that this “linearization query” is not expressible in $\text{FO}+\text{POLY}$. The $\text{FO}+\text{POLY}+\text{TC}$ construction will be presented in Chapter 5 (with preparations in Chapter 4).

3.1 Recursive Functions on the Natural Numbers

We first show that $\text{FO}+\text{LIN}+\text{TCS}$ is computationally complete on the set of natural numbers \mathbf{N} .

Lemma 3.1.1. *For every partial computable function $f : \mathbf{N}^k \rightarrow \mathbf{N}$ there exists a formula $\varphi_f(y)$ in FO+LIN+TCS over the schema $\mathcal{S} = \{S\}$, with S a k -ary relation, such that for any database D over \mathcal{S} such that $S^D = \{(n_1, \dots, n_k)\}$, we have that $\varphi_f(D)$ is defined if and only if $f(n_1, \dots, n_k)$ is defined, and in this case $\varphi_f(D) = \{f(n_1, \dots, n_k)\}$.*

Proof. We show this by simulating the run of a non-deterministic p -counter machine M_f which computes f . Here $M_f = (Q, \delta, q_0, q_f)$ where Q is a finite set of internal states, $q_0 \in Q$ is the initial state, and $q_f \in Q$ is the final (halting) state. δ is a set of quadruples of the form $[q, i, s, q'] \in Q \times \{1, \dots, p\} \times \{Z, P\} \times Q$ or $[q, i, d, q'] \in Q \times \{1, \dots, p\} \times \{-, +\} \times Q$. The quadruple $[q, i, s, q']$ means that if M_f is in state q and the i th counter is equal to zero (when $s = Z$), or positive (when $s = P$), then change the state into q' . The quadruple $[q, i, d, q']$ means that if M_f is in state q , then increase the i th counter by one (when $d = +$) or decrease the i th counter by one (when $d = -$), and change the state into q' . We assume that $Q = \{0, 1, \dots, m-1, m\}$, $q_0 = 0$ and $q_f = m$. Moreover, we assume that $p \geq k$ and that the initial configuration of M_f when computing $f(n_1, \dots, n_k)$ has n_1, \dots, n_k as the values of the first k counters. When a halting state is reached, we assume that the first counter contains $f(n_1, \dots, n_k)$.

We define the first-order formula $\Psi_{\text{step}}(q, n_1, \dots, n_p, q', n'_1, \dots, n'_p)$ as a finite disjunction of the following formulae for $[q, i, s, q']$ and $[q, i, d, q']$ in δ :

$$\begin{aligned} \Psi_{[q,i,Z,q']} &\equiv Q(q) \wedge Q(q') \wedge n'_i = n_i = 0 \wedge \bigwedge_{j \in \{1, \dots, i-1, i+1, \dots, p\}} n_j = n'_j, \\ \Psi_{[q,i,P,q']} &\equiv Q(q) \wedge Q(q') \wedge n'_i = n_i > 0 \wedge \bigwedge_{j \in \{1, \dots, i-1, i+1, \dots, p\}} n_j = n'_j, \\ \Psi_{[q,i,+,q']} &\equiv Q(q) \wedge Q(q') \wedge n'_i = n_i + 1 \wedge \bigwedge_{j \in \{1, \dots, i-1, i+1, \dots, p\}} n_j = n'_j, \\ \Psi_{[q,i,-,q']} &\equiv Q(q) \wedge Q(q') \wedge n'_i = n_i - 1 \wedge \bigwedge_{j \in \{1, \dots, i-1, i+1, \dots, p\}} n_j = n'_j. \end{aligned}$$

The formula Ψ_{step} describes a single step in a run of M_f .

We use the following stop condition σ :

$$\sigma \equiv \exists y_1, \dots, \exists y_p, \exists n_1, \dots, \exists n_k (S(n_1, \dots, n_k) \wedge X(0, n_1, \dots, n_k, \vec{0}_{p-k}, m, y_1, \dots, y_p)).$$

Here, $\vec{0}_\ell$ denotes the ℓ -tuple $(0, \dots, 0)$. The desired formula $\varphi_f(y)$ is

$$\begin{aligned} &\exists y_2, \dots, \exists y_p, \exists n_1, \dots, \exists n_k (S(n_1, \dots, n_k) \\ &\quad \wedge [\text{TC}_{q, \vec{n}; q', \vec{n}'} \Psi_{\text{step}} \mid \sigma](0, n_1, \dots, n_k, \vec{0}_{p-k}, m, y, y_2, \dots, y_p)). \end{aligned}$$

□

3.2 Finite Representation of \mathbf{Z} -linear Constraint Databases

Lemma 3.2.1. *There exists an encoding of \mathbf{Z} -linear constraint databases into finite relational databases over the rationals, and a corresponding decoding, which are both expressible in FO+LIN+TCS.*

Proof. It was shown by Vandeurzen et al. [73, 75] that any \mathbf{Z} -linear constraint database has a finite geometric representation by means of a finite database over \mathbf{Q} consisting of $(n+1)^2$ -ary tuples. Basically, this geometric representation contains the projective coordinates¹ of a complete triangulation of the \mathbf{Z} -linear constraint database. Moreover, this representation can be expressed in FO+POLY. Vandeurzen et al. [73, 75] actually show that this representation can be expressed in an extension of FO+LIN with some limited amount of multiplicative power. Also, the corresponding decoding, which computes the \mathbf{Z} -linear constraint database given its finite geometric representation, can be expressed in this logic.

Hence, the lemma follows, if we can show that FO+LIN+TCS can perform this limited amount of multiplication.

More specifically, we have to be able to express the multiplication of rationals q_i from a finite set $S = \{q_1, \dots, q_m\}$ with a real number x , i.e., $q_i x$ for $i = 1, \dots, m$. First, we express how integers n_i and d_i can be computed in FO+LIN+TCS such that $q_i = \frac{n_i}{d_i}$ for $i = 1, \dots, m$.

We assume that all rational numbers in the set S are positive. The case of negative rational numbers being completely analogous. If both positive and negative rational numbers occur in the set, we separate the positive from the negative and treat both sets separately.

Consider the following mapping $enum$ of $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N} \times \mathbf{N}$:

$$enum : (i, j) \mapsto \begin{cases} (i+1, j-1) & \text{if } j > 0; \\ (0, i+1) & \text{if } j = 0. \end{cases}$$

It is an easy exercise to show for every pair $(p, q) \in \mathbf{N} \times \mathbf{N}$ different from $(0, 0)$, there exists a $k \in \mathbf{N}$, such that $enum^k(0, 0) = (p, q)$. We shall interpret (p, q) as the rational number $\frac{p}{q}$ in case $q \neq 0$, and as 0 otherwise.

Given a rational number q and two natural numbers n and d , we can test in FO+LIN+TCS whether $q = \frac{n}{d}$. This test can be performed as follows. Let $frac : \mathbf{R}^3 \rightarrow \mathbf{R}^3$ be the mapping defined as

$$frac : (q, j, v) \mapsto (q, j-1, v+q).$$

Then $q = \frac{n}{d}$ for $q \in \mathbf{Q}$, and $n, d \in \mathbf{N}$ if and only if $frac^d(q, d, 0) = (q, 0, n)$.

To find the numerator and denominator of a rational number q , we test for each $(n, d) = enum^k(0, 0)$, whether $q = \frac{n}{d}$. For this, we combine $enum$ and $frac$ into a

¹Projective coordinates are used to deal with unbounded databases and the unbounded simplices in their triangulation.

mapping $tryall : \mathbf{R}^5 \rightarrow \mathbf{R}^5$ defined as

$$tryall : (q, i, j, u, v) \mapsto \begin{cases} (q, i, j, u', v') & \text{with } (q, u', v') = frac(q, u, v), \text{ if } u \geq 1, \\ (q, i', j', j', 0) & \text{with } (i', j') = enum(i, j), \text{ if } u = 0. \end{cases}$$

It is clear that there exists an FO+LIN formula, $\psi_{tryall}(q, i, j, u, v, q', i', j', u', v')$, expressing that $tryall(q, i, j, u, v) = (q', i', j', u', v')$. Let $\Psi(q, i, j, u, v, q', i', j', u', v')$ be the formula:

$$q \geq 0 \wedge i \geq 0 \wedge j \geq 0 \wedge i' \geq 0 \wedge j' \geq 0 \wedge q = q' \wedge \psi_{tryall}(q, i, j, u, v, q', i', j', u', v').$$

Given a finite set of rational numbers $S = \{q_1, \dots, q_m\}$, we obtain a denominator and numerator for all these numbers by taking the transitive closure

$$[TC_{q,i,j,u,v;q',i',j',u',v'} \Psi \mid \sigma](\vec{s}, \vec{t}), \quad (3.2.1)$$

where \vec{s} and \vec{t} are 5-tuples of variables, and where

$$\sigma \equiv \forall q(S(q) \rightarrow \exists n \exists d X(q, 0, 0, 0, 0, q, n, d, 0, n)).$$

This condition stops the computation of the transitive closure of (3.2.1) when for each rational number q in S , a pair of natural numbers (n, d) is encountered, such that $dq = n$. If multiple pairs (n, d) represent the same rational number in S , we select the pair with the smallest value of n . Thus, we obtain for each $q \in S$ a unique denominator and numerator.

We now show how to express the multiplication of a finite number of rational numbers with a real number. Without loss of generality, we may assume that the rational number is represented as a numerator/denominator pair, i.e., we may assume that $S = \{(n_1, d_1), \dots, (n_m, d_m)\}$.

Let \max be the largest natural number occurring in S . We now compute any multiplication of the form rn with $r \in \mathbf{R}$, and $0 \leq n \leq \max$ and $n \in \mathbf{N}$.

For this, we consider the following formula

$$\begin{aligned} \mathbf{natmult}(x, y, z, x', y', z') &\equiv x = x' \wedge y' = y - 1 \wedge z' = z + x \\ &\wedge \exists \max(\exists n(S(\max, n) \vee S(n, \max)) \wedge \forall n \forall (S(n, d) \rightarrow s \leq \max \wedge d \leq \max) \\ &\quad \wedge 0 \leq y \wedge y \leq \max). \end{aligned}$$

Then the formula

$$\mathbf{mult}(a, b, c) \equiv [TC_{x,y,z;x',y',z'} \mathbf{natmult}](a, b, 0, a, 0, c)$$

holds if and only if $ab = c$, for $a \in \mathbf{R}$, $b \in \mathbf{N}$ and $b \leq \max$. In this way, we can retrieve any multiple (up to \max) of any real number.

Finally, we define $\mathbf{ratmult}(z, y, n, d) \equiv \exists u(\mathbf{mult}(z, d, u) \wedge \mathbf{mult}(y, n, u))$. This formula holds for (z, y, n, d) if and only if $z = yq$ with $z, y \in \mathbf{R}$, and $q = \frac{n}{d}$ with $(n, d) \in S$. \square

3.3 Natural Number Representation

Lemma 3.3.1. *There exists an encoding of finite relations over the rational numbers into single natural numbers, and a corresponding decoding, which are both expressible in FO+LIN+TCS.*

Proof. We assume that the relation to be encoded involves positive rational numbers only. The general case can be dealt with by splitting the relation into “sign-homogeneous” pieces, dealing with each piece separately, and encoding the tuple of natural numbers obtained for each piece again into a single natural number.

In the proof of Lemma 3.2.1, we have seen that we can go in FO+LIN+TCS from rational numbers (out of a finite set) to denominator/nominator pairs and back. Hence, we can actually assume that the relation to be encoded involves positive natural numbers only.

We will encode this in two steps. In the first step, we encode a finite relation over \mathbf{N} into a finite subset of \mathbf{N} . In the second step, we encode a finite subset of \mathbf{N} into a single natural number. Since queries can be composed, we can treat these two encoding steps (and their corresponding decoding steps) separately.

Encoding, first step A finite k -ary relation s over \mathbf{N} can be encoded into a finite subset $\text{Enc}_1(s)$ of \mathbf{N} :

$$\text{Enc}_1(s) := \left\{ \prod_{i=1}^k p_i^{n_i} \mid (n_1, \dots, n_k) \in s \right\}.$$

Here, p_i denotes the i th prime number.

Now let S be a k -ary relation name. We will construct an FO+LIN+TC formula ϵ_1 over $\{S\}$ such that for any database D where S^D is finite and involves natural numbers only, $\epsilon_1(D) = \text{Enc}_1(S^D)$. For notational simplicity, we give the construction only for the case $k = 2$; the general case is analogous.

Consider the following formula $\psi(x_1, x_2, y, x'_1, x'_2, y')$:

$$\begin{aligned} & \exists u_1 \exists u_2 (S(u_1, u_2) \wedge x_1 \leq u_2 \wedge x_2 \leq u_2) \\ & \wedge ((x_1 > 0 \wedge x'_1 = x_1 - 1 \wedge x'_2 = x_2 \wedge y' = 2y) \\ & \quad \vee (x_1 \leq 0 \wedge x_2 > 0 \wedge x'_1 = 0 \wedge x'_2 = x_2 - 1 \wedge y' = 3y)). \end{aligned}$$

Here, $y' = 2y$ is an abbreviation for $y' = y + y$, and similarly for $y' = 3y$; note that 2 and 3 are the first two prime numbers. Then the desired formula $\epsilon_1(y)$ is

$$\exists n_1 \exists n_2 (S(n_1, n_2) \wedge [\text{TC}_{x_1, x_2, y; x'_1, x'_2, y'} \psi](n_1, n_2, 1, 0, 0, y)).$$

Decoding, first step Let S be a unary relation name. We will construct an FO+LIN+TC formula δ_1 over $\{S\}$ such that for any database D where S^D equals $\text{Enc}_1(r)$ for some r , we have $\delta_1(D) = r$. As above we keep with the case $k = 2$.

Consider now the following formula $\psi(x_1, x_2, y, x'_1, x'_2, y')$:

$$x_1 \geq 0 \wedge x_2 \geq 0 \wedge y \geq 1 \wedge ((x'_1 = x_1 + 1 \wedge x'_2 = x_2 \wedge y' = 2y) \\ \vee (x'_1 = x_1 \wedge x'_2 = x_2 + 1 \wedge y' = 3y)) \wedge \exists u(S(u) \wedge y' \leq u)$$

Then the desired formula $\delta_1(n_1, n_2)$ is

$$\exists u(S(u) \wedge [\text{TC}_{x_1, x_2, y; x'_1, x'_2, y'} \psi](0, 0, 1, n_1, n_2, u)).$$

Encoding, second step A finite ordered subset $s = \{n_1, \dots, n_\ell\}$ of \mathbf{N} can be encoded into a single natural number $\text{Enc}_2(s) := \prod_{i=1}^{\ell} p_i^{n_i}$. Let S be a unary relation name. We will construct an FO+LIN+TCS formula ϵ_2 over $\{S\}$ such that for any database D where S^D is a finite subset of \mathbf{N} , we have $\epsilon_2(D) = \{\text{Enc}_2(S^D)\}$.

We will use the following auxiliary FO+LIN+TCS formulae; we will explain how to get them later (except for *min* and *max* which are easy to get).

- Formulae **card**, **min**, and **max** over $\{S\}$, with the property that for any D where S^D is finite of cardinality ℓ : $\text{card}(D) = \{\ell\}$; $\text{min}(D) = \{\min S^D\}$; and $\text{max}(D) = \{\max S^D\}$.
- Formulae **prime**, **mult**, and **nat**, over $\{M\}$, with M a unary relation name, with the property that for any D where $M^D = \{m\}$ is a natural number singleton:
 - **prime**(D) = $\{p_m\}$;
 - **mult**(D) = $\{(x, y, z) \in \mathbf{R}^3 \mid xy = z \ \& \ y \in \mathbf{N} \ \& \ y \leq m\}$; and
 - **nat**(D) = $\{0, 1, 2, \dots, m\}$.
- Formula **pow** over $\{M, M_2\}$, with M, M_2 unary relation names, with the property that for any D where $M^D = \{m\}$ and $M_2^D = \{m_2\}$ are natural number singletons: $\text{pow}(D) = \{(x, y, z) \in \mathbf{R}^3 \mid x^y = z \ \& \ x \in \mathbf{N} \ \& \ x \leq m \ \& \ y \in \mathbf{N} \ \& \ y \leq m_2\}$.

Using composition, we also obtain:

- **maxprime** \equiv **prime**(**card**), defining p_ℓ where ℓ is the cardinality of S ;
- **nat'** \equiv **nat**(**maxprime**), defining $\{0, 1, 2, \dots, p_\ell\}$; and
- **pow'** \equiv **pow**(**maxprime**, **max**), defining exponentiation of natural numbers $\leq p_\ell$ by natural numbers $\leq \max S$.

We furthermore construct the following formulae:

- **mult'**, obtained from **mult** by replacing each occurrence of a subformula $M(u)$ by

$$\exists p_\ell \exists m(\text{maxprime}(p_\ell) \wedge \text{max}(m) \wedge \text{pow}'(p_\ell, m, u))$$

This formula defines multiplication by natural numbers $\leq p_\ell^{\max S}$.

- **isprime**(p), which defines $\{p_1, p_2, \dots, p_\ell\}$:

$$\mathbf{nat}'(p) \wedge p > 1 \wedge \neg \exists u \exists v (\mathbf{nat}'(u) \wedge \mathbf{nat}'(v) \wedge u > 1 \wedge v > 1 \wedge \mathbf{mult}'(u, v, p)).$$

Consider now the following formula $\psi(x, p, y, x', p', y')$:

$$S(x) \wedge \mathbf{succ}(x, x') \wedge \mathbf{next}(p, p') \wedge \exists y'' (\mathbf{pow}'(p, x, y'') \wedge \mathbf{mult}'(y, y'', y')),$$

where $\mathbf{succ}(x, x')$ is the formula

$$\begin{aligned} & (\neg \mathbf{max}(x) \wedge S(x') \wedge x < x' \\ & \quad \wedge \neg \exists x'' (S(x'') \wedge x < x'' < x')) \vee (\mathbf{max}(x) \wedge x' = x + 1), \end{aligned}$$

and $\mathbf{next}(p, p')$ is the formula

$$\begin{aligned} & (\neg \mathbf{maxprime}(p) \wedge \mathbf{isprime}(p') \wedge p < p' \\ & \quad \wedge \neg \exists p'' (\mathbf{isprime}(p'') \wedge p < p'' < p')) \vee (\mathbf{maxprime}(p) \wedge p' = p + 1). \end{aligned}$$

Then the desired formula $\epsilon_2(n)$ is

$$\begin{aligned} & \exists n_1 \exists m \exists p_\ell (\mathbf{min}(n_1) \wedge \mathbf{max}(m) \wedge \mathbf{maxprime}(p_\ell) \\ & \quad \wedge [\mathbf{TC}_{x,p,y;x',p',y'} \psi](n_1, 2, 1, m + 1, p_\ell + 1, n)). \end{aligned}$$

It remains to show how the auxiliary formulae can be constructed.

Formula $\mathbf{card}(\ell)$ can be written as

$$\begin{aligned} & \exists n_1 \exists m (\mathbf{min}(n_1) \wedge \mathbf{max}(m) \\ & \quad \wedge [\mathbf{TC}_{x,c;x',c'} S(x) \wedge \mathbf{succ}(x, x') \wedge c' = c + 1](n_1, 0, m + 1, \ell)), \end{aligned}$$

where $\mathbf{succ}(x, x')$ is as above.

From the computationally completeness of FO+LIN+TCS (Lemma 3.1.1), we derive directly the **prime**.

For formula **mult**, consider the following formula $\psi(x, y, u, x', y', u')$:

$$x' = x \wedge y' = y - 1 \wedge u' = u + x \wedge 0 < y \wedge \exists m (M(m) \wedge y \leq m)$$

Then $\mathbf{mult}(x, y, z)$ is $[\mathbf{TC}_{x,y,u;x',y',u'} \psi](x, y, 0, x, 0, z)$.

Formula $\mathbf{nat}(n)$ can be written as

$$n = 0 \vee [\mathbf{TC}_{x;x'} 0 \leq x \wedge \exists m (M(m) \wedge x < m) \wedge x' = x + 1](0, n).$$

Finally, for formula **pow**, consider the following formula $\psi(x, u, v, x', u', v')$:

$$\mathbf{nat}(x) \wedge \exists m (M(m) \wedge x < m) \wedge 0 \leq u \wedge \exists m_2 (M_2(m_2) \wedge u < m_2) \wedge u' = u + 1 \wedge \mathbf{mult}(v, x, v')$$

Then $\mathbf{pow}(x, y, z)$ is $(y = 0 \wedge z = 1) \vee [\mathbf{TC}_{x,u,v;x',u',v'} \psi](x, 0, 1, x, y, z)$.

Decoding, second step Let E be a unary relation name. We will construct an FO+LIN+TCS formula δ_2 over $\{E\}$ such that for any database D where E^D is a singleton $\{e\}$ such that e equals $Enc_2(s)$ for some s , we have $\delta_2(D) = s$.

By Lemma 3.1.1, we have formulae **highprime** and **highexp** over $\{E\}$ such that for any D as above, we have **highprime**(D) = $\{p_\ell\}$ and **highexp**(D) = $\{m\}$, where p_ℓ is the highest prime factor of e , and m is the highest exponent of a prime number in the prime factorization of n . Composing the formula **pow** of above with these two formulae, we obtain a formula defining exponentiation of natural numbers $\leq p_\ell$ by natural numbers $\leq m$, which we again denote by **pow'**. Also, analogously to the way we constructed the formula **isprime** of above, we obtain a formula defining $\{p_1, p_2, \dots, p_\ell\}$, which we again denote by **isprime**.

Consider further the following formula $\psi(u, v, u', v')$:

$$0 \leq u \wedge \exists e(E(e) \wedge u \leq e) \wedge v \geq 1 \wedge v' = v \wedge u' = u - v$$

and let **divisor**(d) be the formula

$$\exists e(E(e) \wedge [\text{TC}_{u,v,u',v'}\psi](e, d, 0, d)).$$

Then the desired formula $\delta_2(n)$ is

$$\begin{aligned} \exists p(\text{isprime}(p) \wedge \exists d(\text{pow}'(p, n, d) \wedge \text{divisor}(d)) \\ \wedge \neg \exists n' \exists d'(\text{pow}'(p, n', d') \wedge \text{divisor}(d') \wedge n' > n)). \end{aligned}$$

□

3.4 Completeness Result for \mathbf{Z} -linear Constraint Databases

Theorem 3.4.1. *For every computable query Q on \mathbf{Z} -linear constraint databases, there exists an FO+LIN+TCS formula φ such that for each database D , $\varphi(D)$ is defined if and only if $Q(D)$ is, and in this case $\varphi(D)$ and $Q(D)$ are equal.*

Proof. The proof follows directly from the lemmas above, as is illustrated in the following diagram. Let D be a \mathbf{Z} -linear constraint database, and Q an arbitrary computable query.

$$\begin{array}{ccccc} D & \xrightarrow[\text{database (Lemma 3.2.1)}]{\text{encoding in finite}} & D_{\text{fin}} & \xrightarrow[\text{(Lemma 3.3.1)}]{\text{encoding in integer}} & n_D \in \mathbf{N} \\ Q \downarrow & & & & f_Q \downarrow \text{(Lemma 3.1.1)} \\ Q(D) & \xleftarrow[\text{database (Lemma 3.2.1)}]{\text{decoding from finite}} & Q(D)_{\text{fin}} & \xleftarrow[\text{(Lemma 3.3.1)}]{\text{decoding from integer}} & n_{Q(D)} \end{array}$$

First, D is encoded in a finite database D_{fin} which in its turn is encoded in a natural number n_D . Since Q is computable, there exists a partial computable function f_Q

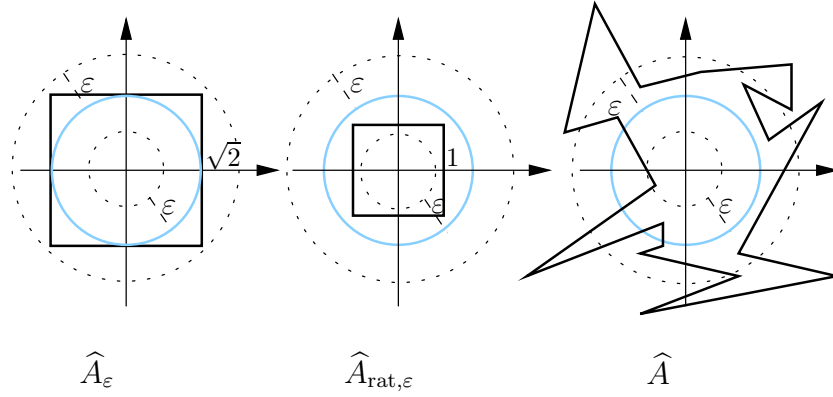


Figure 3.1: Example of an algebraic ε -approximation (left), a rational ε -approximation (middle), and an algebraic linearization (left)

which implements Q on these encodings. Let $n_Q(D)$ be the result of f_Q on input n_D . This integer is decoded into a finite relation $Q(D)_{\text{fin}}$ which in its turn can be decoded in a \mathbf{Z} -linear constraint database D' . This database is then the result of the query Q on the input database D , i.e., $D' = Q(D)$. \square

3.5 Implications for Polynomial Constraint Databases

For polynomial constraint databases we have to settle for less. Although finite representations of polynomial constraint databases exists, it is not known whether a finite encoding can be expressed in FO+POLY+TCS.

Let A be a semi-algebraic set in \mathbf{R}^n . An *algebraic linearization* of A is an \mathbf{A} -linear set \hat{A} in \mathbf{R}^n , such that A and \hat{A} are topologically equivalent. A *rational linearization* of A is a \mathbf{Z} -linear set \hat{A}_{rat} in \mathbf{R}^n , such that A and \hat{A}_{rat} are topologically equivalent.

Let $\vec{x} \in \mathbf{R}^n$, then we denote the *Euclidean norm* $\sqrt{x_1^2 + \dots + x_n^2}$ by $\|\vec{x}\|$. A linearization approximates the set A also from a metric point of view if the following condition is satisfied: for every point \vec{p} in A , $\|\vec{p} - h(\vec{q})\| < \varepsilon$ for a fixed $\varepsilon > 0$, were h is a homeomorphism of \mathbf{R}^n , such that $h(A) = \hat{A}$. If this condition is satisfied for a (rational) linearization, we call this linearization a *(rational) ε -approximation* of the set A . We will denote the rational and algebraic ε -approximation respectively by $\hat{A}_{\text{rat},\varepsilon}$ and \hat{A}_ε .

Example 3.5.1. Consider the planar semi-algebraic set $A = \{(x, y) \in \mathbf{R}^2 \mid x^2 + y^2 = \sqrt{2}\}$. Let $\varepsilon = \frac{1}{2}\sqrt{2}$. In Figure 3.1, we have drawn an algebraic ε -approximation $\hat{A}_\varepsilon = \{(x, y) \in \mathbf{R}^2 \mid \max\{|x|, |y|\} = \sqrt{2}\}$, an rational ε -approximation $\hat{A}_{\text{rat},\varepsilon} = \{(x, y) \in \mathbf{R}^2 \mid \max\{|x|, |y|\} = 1\}$, and a linearization \hat{A} which is not an ε -approximation. \square

Algebraic and rational linearizations exist for any polynomial constraint database. This is no longer true for ε -approximations, where the existence is only guaranteed for bounded polynomial constraint databases. Consider e.g., the semi-algebraic set $\{(x, y) \in \mathbf{R}^2 \mid y = x^2\}$. It is easy to see that this parabola cannot be approximated by a finite number of line segments, and hence has no ε -approximation for any $\varepsilon > 0$.

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define an *algebraic (rational) linearization query* Q_{lin} ($Q_{\text{rat-lin}}$), as a query such that $Q_{\text{rat}}(D)$ ($Q_{\text{rat-lin}}(D)$) is an algebraic (rational) linearization of S^D , for any polynomial constraint database D over \mathcal{S} .

Similarly, for any $\varepsilon > 0$, we define an *algebraic (rational) ε -approximation query* Q_ε ($Q_{\text{rat},\varepsilon}$), as a query such that $Q_\varepsilon(D)$ ($Q_{\text{rat},\varepsilon}(D)$) is an algebraic (rational) ε -approximation of S^D , for any polynomial constraint database D over \mathcal{S} .

We now prove some inexpressibility results for FO+POLY.

Theorem 3.5.1. *The query $Q_{\text{rat-lin}}$ is not expressible in FO+POLY.*

Proof. Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We shall prove that if D is a polynomial constraint database over \mathcal{S} , such that S^D is a finite set of real numbers, then there exists no formula $\text{ratlin}(x) \in \text{FO+POLY}$ over \mathcal{S} , such that $\text{ratlin}(D) = Q_{\text{rat-lin}}(D)$. Note that $Q_{\text{rat-lin}}(D)$ is a set of the same cardinality as S^D consisting of rational numbers only.

More specifically, we shall prove that if we suppose that such a formula $\text{ratlin}(x)$ exists, then there exists a number $M \in \mathbf{N}$ such that if $|S^D| > M$, then $|\text{ratlin}(D)| < M$.

We may assume that $\text{ratlin}(x)$ is a safe query, and hence, since safe queries coincide with the class of queries in the polynomial range-restricted form of Benedikt and Libkin [6], there exists a collection of polynomials $P = \{p_1(x, y_1, \dots, y_k), \dots, p_m(x, y_1, \dots, y_k)\}$ such that $\text{ratlin}(D) \subseteq \text{Root}_P(D)$. Here,

$$\text{Root}_P(D) = \bigcup_{i=1}^m \bigcup_{\vec{a} \in \text{adom}(D)^k} \{x \in \mathbf{R} \mid p_i(x, \vec{a}) = 0\},$$

where $\text{adom}(D)$ is the active domain of D , i.e., the set of all values that occur in the relations in D , and where the p_i are not identical to zero. One may assume that $m = 1$ by considering the polynomial $p(x, \vec{y}) = \prod_{i=1}^m p_i(x, \vec{y})$. Let \mathcal{Y} be the set of all possible sequences of variables in $\{y_1, \dots, y_k\}$ of length k . We define the polynomial

$$q(x, \vec{y}) := \prod_{\sigma \in \mathcal{Y}} p(x, \vec{y}/\sigma),$$

where \vec{y}/σ means that y_i is replaced with the i th element in σ . For example, when $k = 2$ and $p(x, y_1, y_2) = x - y_1 y_2$, then $q(x, y_1, y_2) = (x - y_1 y_1)(x - y_1 y_2)(x - y_2 y_1)(x - y_2 y_2)$. Hence,

$$\text{Root}_P(D) = \{x \in \mathbf{R} \mid (\exists \vec{a} \in \text{adom}(D)^k)(q(x, \vec{a}) = 0 \ \& \ a_1 < a_2 < \dots < a_k)\}. \quad (3.5.1)$$

Let $\mathbf{R}_{<} := \{(y_1, \dots, y_k) \mid y_1 < y_2 < \dots < y_k\} \subseteq \mathbf{R}^k$, and let Z_ℓ be the set

$$\{(a_1, \dots, a_k) \in \mathbf{R}_{<} \mid q(x, \vec{a}) \text{ has exactly } \ell \text{ distinct real zeros}\}.$$

Now, $\mathbf{R}_{<} = Z_0 \cup \dots \cup Z_{\deg_x q}$, with $\deg_x q$ the degree of $q(x, \vec{y})$. Because $\dim \mathbf{R}_{<} = \max\{\dim Z_0, \dots, \dim Z_{\deg_x q}\}$, there exists a p such that $\dim Z_p = k$. Let $Y \subseteq Z_p$ be an open set of dimension k .²

By the Cell Decomposition Theorem [71, Theorem 2.11], Y can be chosen such that there exist p continuous real-valued functions $\xi_1(\vec{y}) < \dots < \xi_p(\vec{y})$ on Y such that

$$q(\xi_j(\vec{y}), \vec{y}) = 0, \text{ for } j = 1, \dots, p.$$

Let M be any number greater than $\max\{k, p + 1\}$, and consider the FO+POLY formula $\mathbf{ratlin}(D/\vec{z}, x)$ where D/\vec{z} denotes that any occurrence of the relation name $S(t)$ is replaced by the disjunction $\bigvee_{i=1}^M (t = z_i)$, where z_1, \dots, z_M are variables not occurring in φ . The set

$$\bigcup_{D, |D| \leq M} \varphi(D) = \{x \mid \exists z_1, \dots, \exists z_M \mathbf{ratlin}(D/\vec{z}, x)\} \quad (3.5.2)$$

is semi-algebraic and may only attain rational values. Since semi-algebraic sets, consisting of rational values only, are finite sets, (3.5.2) consists of a finite number of rational values $\{q_1, \dots, q_N\}$, such that $0 < q_i < q_{i+1}$ for $i = 1, \dots, N - 1$.

Let $IQ_i =](q_i + q_{i+1})/2, (q_{i+1} + q_{i+2})/2[$ for $i = 1, \dots, N - 2$, and $IQ_0 =]-\infty, (q_1 + q_2)/2[$ and $IQ_{N-1} =](q_{N-1} + q_N)/2, +\infty[$. Then there exists an open set $Y' \subseteq Y$ such that for each function $\xi_j(\vec{y})$ there exists a unique interval IQ , which we denote with I_j , such that

$$\{\xi_j(\vec{y}) \mid \vec{y} \in Y'\} \subseteq I_j. \quad (3.5.3)$$

Let $\vec{p} \in Y'$ with coordinates (p_1, \dots, p_k) , and let $\varepsilon > 0$ such that $B^k(\vec{p}, \varepsilon) \subseteq Y'$. Let D be such that $S^D = \{p_1, \dots, p_k, p_{k+1}, \dots, p_M\}$, with

$$\|(p_1, \dots, p_k) - (p_{M-k+1}, \dots, p_M)\| < \varepsilon.$$

Then $\varphi(D)$ is included in the set defined in (3.5.1). But any sequence of length k of elements of $\{p_1, \dots, p_k, p_{k+1}, \dots, p_M\}$ is in Y' , so by (3.5.3) at most p different rational values can be returned by \mathbf{ratlin} .

This is a contradiction, since $\mathbf{ratlin}(x)$ must be satisfied for $M > p$ rational values. \square

It is an open question whether the previous theorem still holds for Q_{lin} rather than $Q_{\text{rat-lin}}$.

With respect to the ε -approximation query, neither the algebraic, nor the rational version can be expressed in FO+POLY. We restrict ourselves to bounded polynomial constraint databases, because otherwise the result would be trivially true by the remark above.

²The *dimension* of a semi-algebraic set A is defined as $\dim A := \max\{p \in \mathbf{N} \mid A \text{ contains a neighborhood which is topologically equivalent to } \mathbf{R}^p\}$. To the empty set, we assign the dimension -1 .

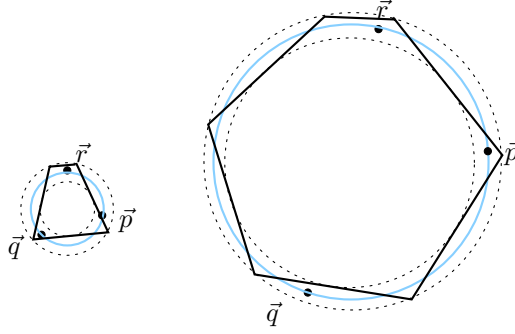


Figure 3.2: When ε is fixed, the number of corner points in the ε -approximation of a circle through three points \vec{p} , \vec{q} , and \vec{r} , increases when the points \vec{p} , \vec{q} , and \vec{r} are moved away from each other.

Proposition 3.5.1. *Let $\varepsilon > 0$ be a real number. The queries Q_ε and $Q_{\text{rat},\varepsilon}$ are not expressible in FO+POLY.*

Proof. Let $\mathcal{S} = \{S\}$, with S a binary relation name. Let D be a polynomial constraint database over \mathcal{S} . We consider the following FO+POLY formulae over \mathcal{S} :

- A formula `circle` such that for any database D over \mathcal{S} , `circle`(D) is either the circle through the points of S^D , if S^D consists of three non-collinear points, or `circle`(D) = \emptyset . This formula is easily seen to be in FO+POLY.
- A formula `cornerpoints` such that for any database D over \mathcal{S} , `cornerpoints`(D) is either the set of points in which S^D is not locally a straight line, in case S^D is semi-linear, or `cornerpoints`(D) = \emptyset , otherwise. By a result of Vandeuren et al. [16], it is expressible in FO+POLY whether a semi-algebraic set is semi-linear. Hence, `cornerpoints` is expressible in FO+POLY.

Assume that the query Q_ε (and similarly, $Q_{\text{rat},\varepsilon}$) is expressible in FO+POLY. Let ε -`approx` be the formula which expresses Q_ε . Then the formula

$$\varphi \equiv \text{cornerpoints}(\varepsilon\text{-approx}(\text{circle}))$$

is also in FO+POLY. However, the number of points in $\varphi(D)$, $|\varphi(D)|$, can be made arbitrarily large by choosing D to be a database over \mathcal{S} such that S^D consist of three points far enough apart (see Figure 3.2). This contradicts the Dichotomy Theorem of Benedikt and Libkin [6], which guarantees the existence of a polynomial p_φ such that $|\varphi(D)| < p_\varphi(|S^D|) = p_\varphi(3)$ in case $|\varphi(D)|$ is finite. \square

In Chapter 5, we show that there exists an FO+POLY+TC expressible algebraic linearization query (Theorem 5.2.1), an FO+POLY+TC expressible rational linearization query (Theorem 5.3.1), an FO+POLY+TC expressible algebraic ε -approximation

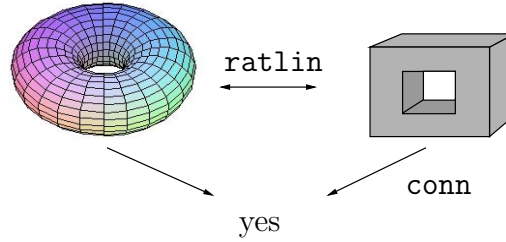


Figure 3.3: The query which asks whether the database is connected, is asked to the linearization of the database.

query (Theorem 5.2.2), and an FO+POLY+TC expressible rational ε -approximation query (Theorem 5.3.2).

We shall denote the FO+POLY+TC formula which expresses the rational linearization by `ratlin`. Let Q be a computable Boolean topological query. Since Q is computable, it is in particular computable on \mathbf{Z} -linear constraint databases, and therefore, by Theorem 3.4.1 expressible on these databases by a formula φ_Q in FO+LIN+TCS.

Because Q is topological, $Q(D)$ is true if and only if $\varphi_Q(\text{ratlin}(D))$ is true (see Figure 3.3). Hence, we have proven the following theorem:

Theorem 3.5.2. *For every computable Boolean topological query Q on polynomial constraint databases, there exists an FO+POLY+TCS formula φ such that for each 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.*

4

Geometric Properties of Semi-algebraic Sets

In this chapter, we discuss a number of geometric properties of semi-algebraic sets, and show that these properties are expressible in $\text{FO}+\text{POLY}$. In Section 4.4, we define the cone radius of a semi-algebraic set in a point and prove that a semi-algebraic set has a cone radius in any of its points. We also define a cone radius query and prove that there exists a cone radius query which is expressible in $\text{FO}+\text{POLY}$. In order to prove this expressibility result, we define the regular and Whitney decomposition of semi-algebraic sets in Section 4.1 and Section 4.2. We also give constructions for both decompositions, and show that these are expressible in $\text{FO}+\text{POLY}$. In Section 4.3, we define the notion of “being in general position” which will be of great importance in the next chapter. In Section 4.5, we define a uniform cone radius decomposition, which will be an important building block in the construction of a linearization. We give a construction of a uniform cone radius decomposition and show that this construction is expressible in $\text{FO}+\text{POLY}$. In Section 4.6 we define box collections and study some properties of them. Finally, we define the box covering query and show that this query is not expressible in $\text{FO}+\text{POLY}$, but is expressible in $\text{FO}+\text{POLY}+\text{TC}$.

We will use the following notation: For $A \subseteq B \subseteq \mathbf{R}^n$, the closure of A in B is denoted by $\text{cl}_B(A)$, and the interior of A in B is denoted by $\text{int}_B(A)$. Similarly, the boundary of A in B is denoted by $\partial_B A = \text{cl}_B(A) - \text{int}_B(A)$. When the ambient space B is \mathbf{R}^n , we omit the subscript B . A closed in

4.1 The Regular Decomposition

In this section, we construct a decomposition of semi-algebraic sets such that a certain regularity condition is satisfied on each part of the decomposition. In order to define this regularity condition, we need to define the tangent space to a semi-algebraic set in a point.

Let A be a semi-algebraic set in \mathbf{R}^n . The secants limit set of A in a point $\vec{p} \in A$, is defined as the set

$$\text{limsec}_{\vec{p}} A := \bigcap_{\eta > 0} \text{cl}(\{\lambda(\vec{u} - \vec{v}) \in \mathbf{R}^n \mid \lambda \in \mathbf{R} \text{ and } \vec{u}, \vec{v} \in A \cap B^n(\vec{p}, \eta)\}).$$

If $\text{limsec}_{\vec{p}} A$ is a vector space (this means that for all $\vec{t}, \vec{t}' \in \text{limsec}_{\vec{p}} A$, also the sum $\vec{t} + \vec{t}'$ is an element of $\text{limsec}_{\vec{p}} A$), then we define the *tangent space of A in \vec{p}* as $\text{T}_{\vec{p}} A := \vec{p} + \text{limsec}_{\vec{p}} A$. If $\text{limsec}_{\vec{p}} A$ is not a vector space, the tangent space of A in \vec{p} is undefined.

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define the query Q_{tangent} as the query such that for any polynomial constraint database D over \mathcal{S} ,

$$Q_{\text{tangent}}(D) := \{(\vec{x}, \vec{v}) \in S^D \times \mathbf{R}^n \mid \text{T}_{\vec{x}} S^D \text{ exists in } \vec{x} \text{ and } \vec{v} \in \text{T}_{\vec{x}} S^D\}.$$

Lemma 4.1.1 ([61]). *The query Q_{tangent} is expressible in FO+POLY.* \square

The set A is *regular in \vec{p}* if and only if there exist a neighborhood U of \vec{p} such that the orthogonal projection of $A \cap U$ on the $\text{T}_{\vec{p}} A$ is bijective. A set is *regular* if it is regular in all its points.

Example 4.1.1. In Figure 4.1 we have illustrated the three possible cases: $\text{T}_{\vec{p}} A$ does not exist, $\text{T}_{\vec{q}} A$ and $\text{T}_{\vec{r}} A$ exist, but A is not regular in \vec{q} and \vec{r} , and finally, A is regular in \vec{s} . \square

We denote the set of points where A is regular and where the local dimension of A is k with $\text{Reg}_k(A)$. Note that $\text{Reg}_k(A)$ is either empty or $\dim \text{Reg}_k(A) = k$.

Define inductively for $k = n, n-1, \dots, 0$, the sets

$$R_k := \text{Reg}_k(A - \bigcup_{j=k+1}^n R_j). \quad (4.1.1)$$

These sets are pairwise disjoint and form a decomposition of A , i.e.,

$$A = R_n \cup R_{n-1} \cup \dots \cup R_0. \quad (4.1.2)$$

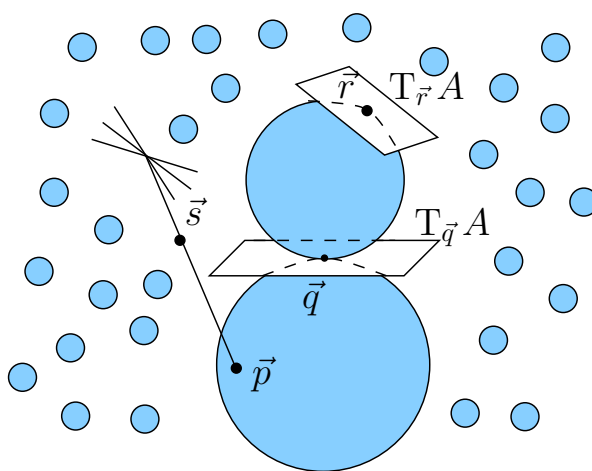


Figure 4.1: A has no tangent space in \vec{p} , A has a tangent space in \vec{q} and \vec{r} , but is not regular in these points, and A is regular in \vec{s} .

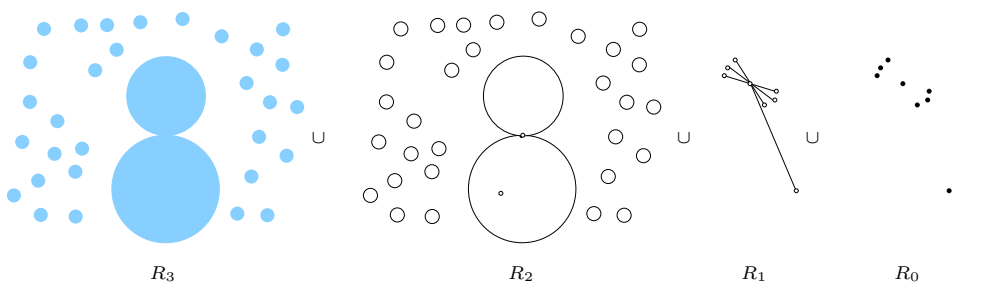


Figure 4.2: The three-dimensional set A of Figure 4.1 is decomposed into four parts R_0, R_1, R_2 , and R_3 according to the construction of the regular decomposition.

Note that $n + 1$ parts are really sufficient, because for any semi-algebraic set $X \subseteq \mathbf{R}^n$ of dimension d , $X - \text{Reg}_d(X)$ has a strict lower dimension than X [72].

Moreover, by (4.1.1) each R_k is regular and hence, we define the *regular decomposition of A* as the $n + 1$ sets R_0, \dots, R_n . In Figure 4.2 we have drawn an example of the regular decomposition of a three-dimensional set in \mathbf{R}^3 .

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define the $n + 1$ queries $Q_{k\text{-reg}}$ inductively as the queries such that for every polynomial constraint database,

$$Q_{k\text{-reg}}(D) := R_k$$

for $k = 0, \dots, n$, with R_0, \dots, R_n the regular decomposition of S^D .

It is proved by Rannou [61], that checking whether a semi-algebraic set is regular in a point is first-order expressible, hence, the next lemma follows.

Lemma 4.1.2. *The queries $Q_{k\text{-reg}}$, $k = 0, 1, \dots, n$ are expressible in FO+POLY. \square*

Regular decompositions of semi-linear sets are fully treated in [16, 73]. It is shown there, that on semi-linear databases, the $n + 1$ queries $Q_{k\text{-reg}}$ are already expressible in FO+LIN. There is however a great difference. Indeed, in the semi-algebraic case, regularity implies that the set is a C^1 -smooth algebraic variety, while in the semi-linear case, regularity implies that the set is a C^∞ -smooth algebraic variety. One could ask if it is possible to define a regularity condition in first-order logic, such that it also induces C^∞ -smoothness of semi-algebraic sets, but this is impossible [78].

4.2 The Whitney Decomposition

In this section, we refine the regular decomposition of a semi-algebraic set, such that any two points in the same part of the regular decomposition, have the same topological type. That this not necessarily holds for regular decompositions is shown in the following example.

Example 4.2.1. Let $A \subseteq \mathbf{R}^3$ be the semi-algebraic set defined by the polynomial constraint formula $x^2 - zy^2$. This set is known as the *Whitney Umbrella*. As shown in Figure 4.3, the regular decomposition consists of two non-empty parts: R_1 and R_2 . It is clear that any two points in R_2 have the same topological type (this will be defined formally in the next section). However, in R_1 , the topological type of the origin is different from that of any other point in R_1 . \square

To avoid such situations as in Example 4.2.1, Whitney introduced the following condition for regular sets X and Y in \mathbf{R}^n , and a point $\vec{x} \in X$ [76, 77]. We say that the triple (X, \vec{x}, Y) has the *Whitney property* when the following holds. If $(\vec{y}_i)_{i \in \mathbf{N}}$ is a sequence of points in Y which converge to \vec{x} such that the sequence of tangent spaces $T_{\vec{y}_i} Y$ converge to a subspace $\tau \subset \mathbf{R}^n$, and if $(\vec{x}_i)_{i \in \mathbf{N}}$ is a sequence of points in X which converges to \vec{x} such that the sequence of lines $(\{\lambda(\vec{x}_i - \vec{y}_i) \mid \lambda \in \mathbf{R}\})_{i \in \mathbf{N}}$ converges to a line $\ell \subset \mathbf{R}^n$, then we must have that $\ell \subset \tau$. We say that the pair (X, Y) has the Whitney property if (X, \vec{x}, Y) has the Whitney property in every point $\vec{x} \in X$.

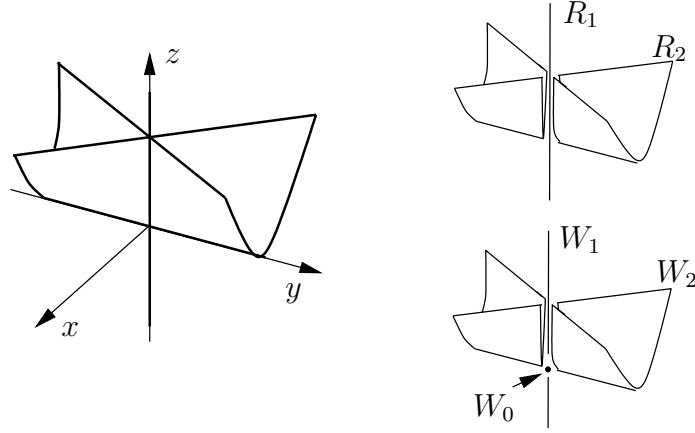


Figure 4.3: The regular decomposition (top right) and the Whitney decomposition (bottom right) of the Whitney Umbrella.

Define inductively for $k = n, n - 1, \dots, 0$, the sets

$$R'_k := \text{Reg}_k(A - \bigcup_{j=k+1}^n W_j) \quad (4.2.1)$$

$$W_k := \bigcap_{j=k+1}^n \text{int}_{R'_k}(\{\vec{p} \in R'_k \mid (R'_k, \vec{p}, W_j) \text{ has the Whitney property}\}). \quad (4.2.2)$$

These set W_0, \dots, W_n are pairwise disjoint and form a decomposition of A , i.e.,

$$A = W_n \cup W_{n-1} \cup \dots \cup W_0. \quad (4.2.3)$$

Note that $n + 1$ parts are really sufficient, because for any two regular semi-algebraic sets $X, Y \subseteq \mathbf{R}^n$, the set of points of X , where (X, \vec{x}, Y) does not have the Whitney property, has a strict lower dimension than X [72].

Moreover, by (4.2.1) each W_k is regular, and by (4.2.2) (W_k, W_j) has the Whitney property for every $j > k$. Hence, we define the *Whitney decomposition of A* as the $n + 1$ sets W_0, \dots, W_n .

Example 4.2.2. In Figure 4.3 we have drawn the Whitney decomposition of the Whitney Umbrella. It consist of three parts W_0, W_1 , and W_2 . As can be easily verified, the topological type remains constant on each connected component of these parts. \square

Let $\mathcal{S} = \{S_1, S_2\}$, with S_1 and S_2 two n -ary relation names. We define the query Q_{Whitney} , as the query such that for every polynomial constraint database D over \mathcal{S} ,

$$Q_{\text{Whitney}}(D) := \{\vec{x} \in \mathbf{R}^n \mid S_1^D, S_2^D \text{ are regular} \\ \text{and } (S_1^D, \vec{x}, S_2^D) \text{ has the Whitney property}\}.$$

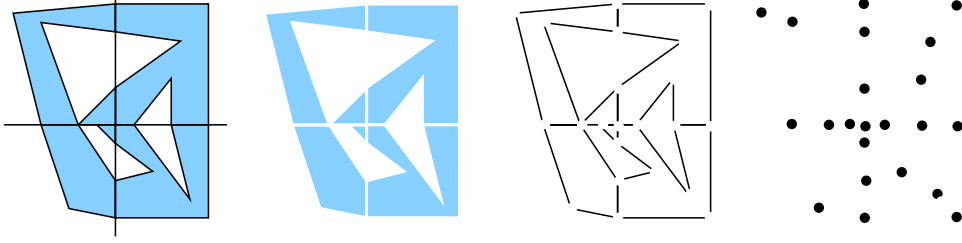


Figure 4.4: An example of the Whitney decomposition compatible with $\mathbf{R}_1 := \{0\} \times \mathbf{R}$, $\mathbf{R}_2 := \mathbf{R} \times \{0\}$, and $\mathbf{R}_{12} := (0, 0)$.

Lemma 4.2.1 ([61]). *The query $Q_{Whitney}$ is expressible in FO+POLY.* \square

Let A be a semi-algebraic set in \mathbf{R}^n . For each $k = 0, \dots, n$, and any sequence of $i_1, \dots, i_k \in \{1, \dots, n\}$ such that $i_1 < i_2 < \dots < i_k$, we define the sets $\mathbf{R}_{i_1 \dots i_k} := \{\vec{x} \in \mathbf{R}^n \mid x_{i_1} = 0, \dots, x_{i_k} = 0\}$.

We now define a Whitney decomposition Z_0, \dots, Z_n of A such that for each connected component Z of the Z_i , either $Z \subseteq \mathbf{R}_{i_1 \dots i_k}$, or $Z \cap \mathbf{R}_{i_1 \dots i_k} = \emptyset$ for any i_1, \dots, i_k . We then say that Z_0, \dots, Z_n is *compatible with* $\{\mathbf{R}_{i_1 \dots i_k}\}$.

Define $A^{\sigma_1 \dots \sigma_n} := \{\vec{x} \in A \mid x_1 \sigma_1 0, \dots, x_n \sigma_n 0\}$, with $\sigma_1, \dots, \sigma_n \in \{<, =, >\}$. For each n -tuple $\sigma = (\sigma_1, \dots, \sigma_n) \in \{<, =, >\}^n$, we define inductively for $k = n, n-1, \dots, 0$, the sets

$$R_k^\sigma := \text{Reg}_k(A^\sigma - \bigcup_{j=k+1}^n Z_j) \quad (4.2.4)$$

$$W_k^\sigma := \bigcap_{j=k+1}^n \text{int}_{R_k^\sigma}(\{\vec{p} \in R_k^\sigma \mid (R_k^\sigma, \vec{p}, Z_j) \text{ has the Whitney property}\}) \quad (4.2.5)$$

$$Z_k^\sigma := W_k^\sigma - \text{cl}\left(\bigcup_{\substack{\sigma' \in \{<, =, >\}^n \\ \sigma' \neq \sigma}} R_k^{\sigma'}\right). \quad (4.2.6)$$

Then we define $Z_k := \bigcup_{\sigma \in \{<, =, >\}^n} Z_k^\sigma$.

These set Z_0, \dots, Z_n are pairwise disjoint and form a decomposition of A , i.e.,

$$A = Z_n \cup Z_{n-1} \cup \dots \cup Z_0. \quad (4.2.7)$$

Moreover, by (4.2.4) each Z_k is regular, by (4.2.5) (Z_k, Z_j) has the Whitney property for every $j > k$, and (4.2.6) ensures that the connected components of Z_k lie either completely in $\mathbf{R}_{i_1 \dots i_k}$, or are disjoint with $\mathbf{R}_{i_1 \dots i_k}$, for any i_1, \dots, i_k . Hence, we define the *Whitney decomposition of A compatible* $\{\mathbf{R}_{i_1 \dots i_k}\}$ as the $n+1$ sets Z_0, \dots, Z_n .

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define the $n+1$ queries $Q_{k\text{-Whitney}}$ inductively as the queries such that for every polynomial constraint database D ,

$$Q_{k\text{-Whitney}}(D) := Z_k$$

for $k = 0, \dots, n$, with Z_0, \dots, Z_n the Whitney decomposition of S^D compatible $\{\mathbf{R}_{i_1 \dots i_k}\}$.

The following lemma is immediately implied by Lemma 4.1.2 and Lemma 4.2.1.

Lemma 4.2.2. *The queries $Q_{k\text{-Whitney}}$, $k = 0, 1, \dots, n$ are expressible in FO+POLY.* \square

4.3 Transversality

In computational geometry, a convenient assumption is the hypothesis of “general position”, which dispenses with the detailed consideration of special cases. In the description of the linearization algorithm in Section 5.2, we would like to assume this hypothesis. However, we need to make precise what we mean by general position, and see if this may be assumed indeed.

Let A and B be two regular semi-algebraic sets in \mathbf{R}^n . We say that A and B intersect transversally at $\vec{p} \in A \cap B$, if ¹

$$\mathbb{T}_{\vec{p}}A + \mathbb{T}_{\vec{p}}B = \mathbf{R}^n. \quad (4.3.1)$$

The sets A and B are *in general position* if they intersect transversally in every point of $A \cap B$. We denote this by $A \pitchfork B$. This is illustrated in Figure 4.5 and Figure 4.6 where some examples of transversal and non-transversal intersections in \mathbf{R}^2 and \mathbf{R}^3 are depicted.

Let $\mathcal{A} = \{A_1, \dots, A_n\}$ and $\mathcal{B} = \{B_1, \dots, B_m\}$ be finite sets of regular semi-algebraic sets in \mathbf{R}^n such that $A_i \cap A_j = \emptyset$ ($B_i \cap B_j = \emptyset$) for $i \neq j$. We say that \mathcal{A} and \mathcal{B} are in general position if A_i and B_j are in general position for every $i = 1, \dots, n$ and every $j = 1, \dots, m$. We denote this with $\mathcal{A} \pitchfork \mathcal{B}$.

Let $\mathcal{S} = \{S_1, S_2\}$, with S_1 and S_2 two n -ary relation names. We define the Boolean query Q_{\pitchfork} , such that for every polynomial constraint database D over \mathcal{S} ,

$$Q_{\pitchfork}(D) := \text{true if and only if } S_1^D \text{ and } S_2^D \text{ are regular, and } S_1^D \pitchfork S_2^D.$$

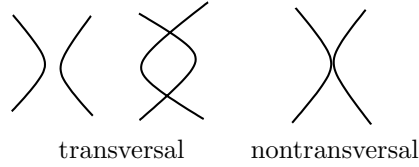
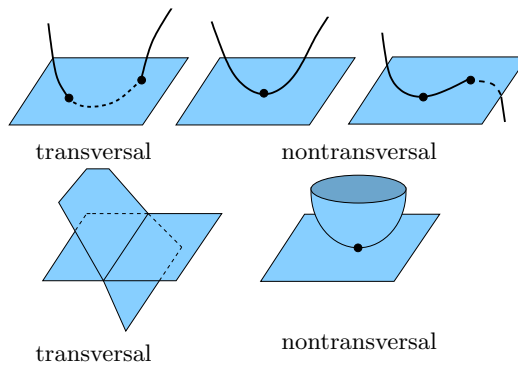
Condition (4.3.1) can be readily expressed in FO+POLY, and by Lemma 4.1.2, regularity is expressible in FO+POLY. Hence:

Lemma 4.3.1. *The Boolean query Q_{\pitchfork} is expressible in FO+POLY.* \square

Given two arbitrary regular semi-algebraic sets A and B in \mathbf{R}^n which are not in general position, we can ask how to force them to be in general position? The following theorem answers this question. A translation of a set $X \subset \mathbf{R}^n$ is a set of the form $X + \tau := \{\vec{x} + \tau \in \mathbf{R}^n \mid \vec{x} \in X\}$, where $\tau \in \mathbf{R}^n$.

Theorem 4.3.1 ([33]). *Let A and B two regular semi-algebraic sets in \mathbf{R}^n . For almost all $\tau \in \mathbf{R}^n$, we have that $A + \tau$ and B are in general position.* \square

¹Let U and V be two subspaces of a vector space X , then the *sum* $U + V$ is the set of all vectors $\vec{u} + \vec{v}$, where $\vec{u} \in U$ and $\vec{v} \in V$. Besides, $U + V$ is a subspace of X .

Figure 4.5: Curves in \mathbf{R}^2 .Figure 4.6: Curves and surfaces in \mathbf{R}^3 .

Here, “almost all” means that the set of translation vectors τ for which $A + \tau$ and B are not in general position has *measure zero*.² Since a set of measure zero cannot contain an open set in \mathbf{R}^n , the set of translation VECTors τ for which $A + \tau$ and B are in general position is dense in \mathbf{R}^n .

Actually, in [33], Theorem 4.3.1 is proved for C^∞ -smooth varieties in \mathbf{R}^n . However, every regular semi-algebraic set admits a decomposition into a finite number of C^∞ -smooth varieties, called the Nash decomposition [8]. So the case of regular semi-algebraic sets can be reduced to the case of C^∞ -smooth varieties.

Moreover, Theorem 4.3.1 can be easily generalized as follows:

Corollary 4.3.1. *Let $\mathcal{A} = \{A_1, \dots, A_n\}$ and $\mathcal{B} = \{B_1, \dots, B_m\}$ be finite sets of regular semi-algebraic sets in \mathbf{R}^n such that $A_i \cap A_j = \emptyset$ ($B_i \cap B_j = \emptyset$) for $i \neq j$. Then for almost all $\tau \in \mathbf{R}^n$, $\mathcal{A} + \tau \pitchfork \mathcal{B}$. \square*

We mention two useful properties of sets in general position: Let \mathcal{A} and \mathcal{B} be as above, then if $\mathcal{A} \pitchfork \mathcal{B}$, then there exists an $\varepsilon > 0$ such that $\mathcal{A} + \tau \pitchfork \mathcal{B}$ for any $\tau \in \mathbf{R}^n$ of norm less than ε . One says that transversality is a *stable* property [33]. A second useful property is that the intersection of two regular sets in general position, is again regular.

²A set in \mathbf{R}^n has *measure zero* if it can be covered by a countable number of n -dimensional boxes with arbitrary small volume.

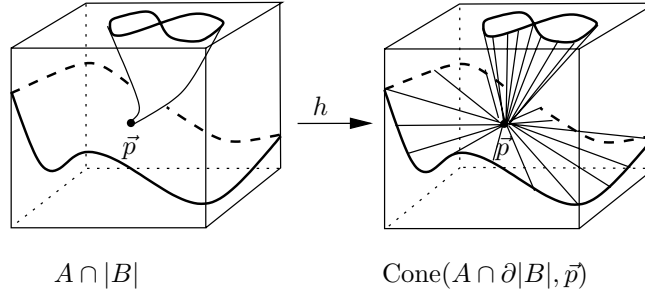


Figure 4.7: The local conic structure of semi-algebraic sets.

4.4 The Cone Radius

Let A be a semi-algebraic set in \mathbf{R}^n , and \vec{p} be a point in \mathbf{R}^n . We define the *cone with base A and top \vec{p}* as the union of all closed line segments between \vec{p} and points in A . Formally, this is the set $= \{t\vec{b} + (1-t)\vec{p} \mid \vec{b} \in A, 0 \leq t \leq 1\}$ and we denote this set with $\text{Cone}(A, \vec{p})$.

Consider a $2n$ -tuple $B = (a_1, b_1, \dots, a_n, b_n) \in \mathbf{R}^{2n}$ with $a_i \leq b_i$ for each i . One can associate with each such tuple an n -ary relation $|B|$ in \mathbf{R}^n :

$$|B| := \{(x_1, \dots, x_n) \in \mathbf{R}^n \mid (a_1 \leq x_1 \leq b_1) \wedge \dots \wedge (a_n \leq x_n \leq b_n)\}.$$

We call B a *box* in \mathbf{R}^n and $|B|$ is the *geometric realization of B* . The *dimension* of a box is the maximal number of pairs (a_i, b_i) with $a_i \neq b_i$. The *diameter* of a box B , $\text{diam}(B)$, equals $(\sum_{i=1}^n (b_i - a_i)^2)^{1/2}$. The *center* of B is the point $(a_1 + (b_1 - a_1)/2, \dots, a_n + (b_n - a_n)/2)$.

We define a *cone radius* of A in a point $\vec{p} \in \text{cl}(A)$ as a positive real number ε such that if B is an n -dimensional box in \mathbf{R}^n with

1. $\vec{p} \in \text{int}(|B|)$; and
2. $|B| \subseteq (p_1 - \varepsilon, p_1 + \varepsilon) \times \dots \times (p_n - \varepsilon, p_n + \varepsilon)$,

then, $A \cap |B|$ is homeomorphic to $\text{Cone}(A \cap \partial|B|, \vec{p})$, in case $\vec{p} \in A$, and $A \cap |B|$ is homeomorphic to $\text{Cone}(A \cap \partial|B|, \vec{p}) - \{\vec{p}\}$, in case $\vec{p} \in \text{cl}(A) - A$ (see Figure 4.7).

Two points \vec{p} and \vec{q} have the same *topological type* (with respect to A), if $A \cap B^n(\vec{p}, \varepsilon_{\vec{p}})$ is topologically equivalent to $A \cap B^n(\vec{q}, \varepsilon_{\vec{q}})$, where $\varepsilon_{\vec{p}}$ ($\varepsilon_{\vec{q}}$) is a cone radius of \vec{p} (\vec{q}).

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define a *cone radius query* Q_{radius} , as a query such that for every polynomial constraint database D over \mathcal{S} ,

$$Q_{\text{radius}}(D) := \{(\vec{p}, r) \in \text{cl}(S^D) \times \mathbf{R} \mid r \text{ is a cone radius of } S^D \text{ in } \vec{p}\},$$

and such that for every $\vec{p} \in \text{cl}(S^D)$ there exists a pair $(\vec{p}, r) \in Q_{\text{radius}}(D)$.

Theorem 4.4.1 (Local Cone Structure). *Let A be a semi-algebraic set in \mathbf{R}^n , and let \vec{p} be a point in $\text{cl}(A)$. Then there exists a cone radius of A in \vec{p} .*

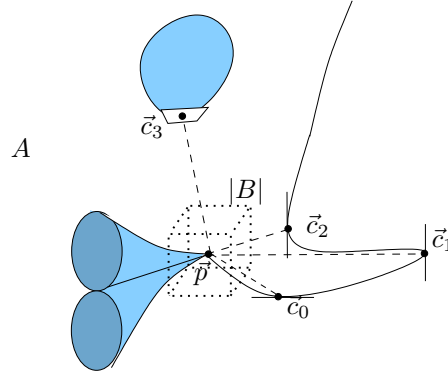


Figure 4.8: The point \vec{c}_0 is a critical point of $f_3(x, y, z) = (p_3 - x_3)^2$, \vec{c}_1 , and \vec{c}_2 are critical points of $f_1(x, y, z) = (p_1 - x_1)^2$, and \vec{c}_3 is a critical point of $f_7(x, y, z) = (p_1 - x_1)^2 + (p_2 - x_2)^2 + (p_3 - x_3)^2$. Since no critical point of the functions f_i for $i = 1, \dots, 7$, lies inside the box $B = (p_1 - \varepsilon, p_1 + \varepsilon, p_2 - \varepsilon, p_2 + \varepsilon, p_3 - \varepsilon, p_3 + \varepsilon)$, every number smaller than ε is a cone radius of A in \vec{p} .

Proof. Let Z_0, \dots, Z_n be the Whitney decomposition of A compatible with $\{\mathbf{R}_{i_1 \dots i_k}\}$. Let $\vec{x} = (x_1, \dots, x_n) \in \mathbf{R}^n$, and define $f_1(\vec{x}) = (p_1 - x_1)^2, \dots, f_n(\vec{x}) = (p_n - x_n)^2, f_{n+1}(\vec{x}) = (p_1 - x_1)^2 + (p_2 - x_2)^2, \dots, f_m(\vec{x}) = (p_{n-1} - x_{n-1})^2 + (p_n - x_n)^2, f_{m+1}(\vec{x}) = (p_1 - x_1)^2 + (p_2 - x_2)^2 + (p_3 - x_3)^2, \dots, f_\ell(\vec{x}) = (p_1 - x_1)^2 + \dots + (p_n - x_n)^2$.

Let $\varepsilon > 0$ and let E be an n -dimensional box $(p_1 - \varepsilon, p_1 + \varepsilon, \dots, p_n - \varepsilon, p_n + \varepsilon)$, such that for $i = 0, \dots, \ell$ and $k = 0, \dots, n$, the restrictions

$$f_i|((Z_k \cap |E|) - f_i^{-1}(0)) \rightarrow \mathbf{R} \text{ have no critical points.} \quad (4.4.1)$$

Here, the *critical points* of $f_i|Z_k$ are the points $\vec{x} \in Z_k$ where the differential mapping $d_{\vec{x}}(f_i|Z_k)$ is not surjective.

Claim 4.4.1. *There exists a positive real number ε such that (4.4.1) holds.*

Proof of Claim. A *critical value* of $f_i|Z_k$ is the image by $f_i|Z_k$ of a critical point. The set of critical points of $f_i|Z_k$ is semi-algebraic and admits a C^1 -cell decomposition $\mathcal{C} = \{C_1, \dots, C_m\}$ such that $f_i|C_j$ is C^1 [71]. Sard's Theorem for C^1 -mapping [79] implies that each $f_i|C_j$ attains only a finite number of values. Hence the image by $f_i|Z_k$ of the set of critical points is finite. This is true for every $i = 1, \dots, \ell$ and every $k = 0, \dots, n$. Denote with r_{ik} the minimal positive critical value of $f_i|Z_k$. (If this not exists, set $r_{ik} = 1$.) Then any $\varepsilon < \min\{\sqrt{r_{ik}/n} \mid i = 1, \dots, \ell, k = 0, \dots, n\}$ is such that (4.4.1) is satisfied. \square

We can actually give a geometric interpretation of the critical points.

Claim 4.4.2. *A point $\vec{x} \in \mathbf{R}^n$ is a critical point of $f_i|Z_k$, with $f_i = (p_{i_1} - x_{i_1})^2 + \dots + (p_{i_m} - x_{i_m})^2$, if and only if the tangent space of Z_k in \vec{x} is orthogonal to $(0, \dots, p_{i_1} - x_{i_1}, \dots, p_{i_m} - x_{i_m}, \dots, 0)$.*

Proof of Claim. We assume that $f_i = x_1^2 + \cdots + x_m^2$, the other cases being similar. We compute the differential $d_{\vec{x}}(f_i|Z_k)$ as follows: Locally around \vec{x} , we may assume that the projection on the first k coordinates $\Pi : Z \rightarrow U \subset \mathbf{R}^k$, is a homeomorphism. By definition of the differential, $d_{\vec{x}}(f_i|Z_k) = (d_{(x_1, \dots, x_k)}g)(d_{(x_1, \dots, x_k)}\Pi^{-1})^{-1}$, where $g = (f_i|Z_k) \circ \Pi^{-1}$. By the C^1 Inverse Function Theorem, we may assume that $\Pi^{-1} : U \rightarrow Z : (x_1, \dots, x_k) \mapsto (x_1, \dots, x_k, \varphi_{k+1}, \dots, \varphi_n)$, where $\varphi_i(x_1, \dots, x_k)$ are C^1 -mappings, and hence $g : U \mapsto \mathbf{R} : (x_1, \dots, x_k) = \sum_{i=1}^k (p_i - x_i)^2 + \sum_{j=k+1}^m (p_j - \varphi_j(x_1, \dots, x_k))^2$. An elementary calculation shows that the differential of $f_i|Z_k$ in \vec{x} is the vector

$$d_{\vec{x}}(f_i|Z_k) = 2 \left(\left((p_i - x_i) + \sum_{j=k+1}^m (p_j - x_j) \frac{\partial \varphi_j}{\partial x_i}(x_1, \dots, x_k) \right)_{i=1, \dots, k}, \underbrace{0, \dots, 0}_{n-k \text{ times}} \right).$$

Since $d_{(x_1, \dots, x_k)}\Pi^{-1}$ is an isomorphism between the tangent space $T_{(x_1, \dots, x_k)}U$ of U in the projection $\Pi(\vec{x})$, and the tangent space $T_{\vec{x}}Z$ of Z in \vec{x} , any tangent vector $(v_1, \dots, v_n) \in T_{\vec{x}}Z$ is of the form $(d_{(x_1, \dots, x_k)}\Pi^{-1})(v_1, \dots, v_k)$. More specifically, any tangent vector $\vec{v} \in T_{\vec{x}}Z$ can be written as

$$(v_1, \dots, v_n) = (v_1, \dots, v_k, \sum_{i=1}^k \frac{\partial \varphi_{k+1}}{\partial x_i}(x_1, \dots, x_k)v_i, \dots, \sum_{i=1}^k \frac{\partial \varphi_n}{\partial x_i}(x_1, \dots, x_k)v_i).$$

Hence, the product

$$d_{\vec{x}}(f_i|Z_k)\vec{v} = 2 \sum_{i=1}^k (p_i - x_i)v_i + 2 \sum_{j=k+1}^m (p_j - x_j) \left(\sum_{i=1}^k \frac{\partial \varphi_j}{\partial x_i}(x_1, \dots, x_k)v_i \right),$$

is equal to $2 \sum_{i=1}^m (x_i - p_i)v_i$. This implies that the differential mapping $d_{\vec{x}}(f_i|Z_k)$ is not surjective if and only if $2 \sum_{i=1}^m (p_i - x_i)v_i = 0$ for all tangent vectors $\vec{v} \in T_{\vec{x}}Z_k$. \square

The proof of the theorem now continues as follows. If (4.4.1) is satisfied, then for any n -dimensional box $B = (p_1 - a_1, p_1 + b_1, \dots, p_n - a_n, p_n + b_n)$, such that $|B| \subseteq |E|$ there exists a homeomorphism

$$h : A \cap |B| \rightarrow \text{Cone}(A \cap \partial|B|, \vec{p}).$$

The homeomorphism h is obtained as follows: By the construction of the Whitney decomposition Z_0, \dots, Z_k , each Z_i is the disjoint union of Z_i^σ for $\sigma \in \{<, =, >\}^n$.

For each Z_i^σ , set

$$\xi_i^\sigma = -\text{grad}(f_j|Z_i^\sigma).$$

Here f_j is chosen so that if $\mathbf{R}_{i_1 \dots i_k}$ is the smallest set (in dimension) such that $Z_i^\sigma \subseteq \mathbf{R}_{i_1 \dots i_k}$, then $f_j(\vec{x}) = x_{i_1'}^2 + \cdots + x_{i_{n-k}'}^2$ with $i_1', \dots, i_{n-k}' \in \{1, \dots, n\} - \{i_1, \dots, i_k\}$ (see Figure 4.9). Recall that the *gradient vector field* $\text{grad}(f|X)$ is defined as a mapping from $X \rightarrow \bigcup_{\vec{x} \in X} T_{\vec{x}}X$ such that $(d_{\vec{x}}f|X)\vec{w} = \text{grad}(f|X)(\vec{x})\vec{w}$ for all vectors

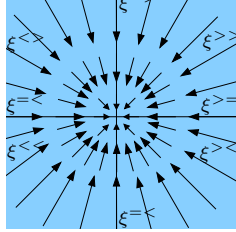


Figure 4.9: The vector fields ξ_i^σ for the case that $A = \mathbf{R}^2$ and $\vec{p} = (0, 0)$. Here, $\xi^{>=} = \xi^{<=} = (-2x, 0)$, $\xi^{=<} = \xi^{>} = (0, -2y)$, and $\xi^{>>} = \xi^{<<} = \xi^{<>} = \xi^{<<} = (-2x, -2y)$.

$\vec{w} \in T_{\vec{x}} X$. Clearly $\text{grad}(f|X)(\vec{x}) = 0$ if and only if \vec{x} is a critical point of $f|X$. On each stratum Z_i^σ , we can obtain a continuous flow θ_i^σ , i.e., a continuous mapping $I \times (Z_i^\sigma \cap \partial|B|) \rightarrow Z_i^\sigma \cap |B|$ with $I \subset \mathbf{R}$, such that $\theta_i^\sigma(0, \vec{x}_0) = \vec{x}_0$, and

$$\frac{d\theta_i^\sigma(t, \vec{x}_0)}{dt} = \xi_i^\sigma(\theta_i^\sigma(t, \vec{x}_0)),$$

for all $\vec{x}_0 \in Z_i^\sigma \cap \partial|B|$. Moreover, by (4.4.1), the vector field $\xi_i^\sigma = -\text{grad}(f_j|Z_k)$ is nowhere vanishing on $Z_i \cap |B|$ and points to a direction of decreasing f_j .

In general we cannot expect to obtain a continuous flow on the set A by just putting the flows θ_i^σ together. However, by (4.4.1), the Whitney decomposition of $|B|$ compatible with $\{\mathbf{R}_{i_1 \dots i_k}\}$ is transversal with Z_0, \dots, Z_n . Hence, we obtain a Whitney decomposition Y_0, \dots, Y_n of $A \cap |B|$ compatible with $\{\mathbf{R}_{i_1 \dots i_k}\}$ (Lemma 1.3, [26]). By Lemma I.1.3, [66], there exists a controlled tube system T_0, \dots, T_n of Y_0, \dots, Y_n . By Lemma I.1.5, [66], we can use this tube system to modify the vector fields ξ_i^σ inductively on the dimension of Y_i such that

1. the resulting vector fields point to a direction of decreasing f_j (for the appropriate f_j); and
2. the flows of the resulting vector fields can be put together to get a continuous flow $\Theta : I \times (A \cap \partial|B|) \rightarrow A \cap |B|$.

For each $t \in (0, 1]$, define the box $B_t = (p_1 + t(a_1 - p_1), p_1 + t(b_1 - p_1), \dots, p_n + t(a_n - p_n), p_n + t(b_n - p_n))$. Because for each point $\vec{x}_0 \in A \cap \partial|B|$, the flow $\Theta(t, \vec{x}_0)$ has no tangent line parallel to the coordinate axis (Claim 4.4.2), and approaches \vec{p} for increasing values of t , we obtain the homeomorphisms

$$h_t : A \cap \partial|B| \rightarrow (A \cap |B|) - \{\vec{p}\} : \vec{x} \mapsto \Theta(\vec{x}) \cap \partial|B_t|,$$

where $\Theta(\vec{x}) = \{\vec{x}' \in A \cap |B| \mid \exists t \Theta(t, \vec{x}) = \vec{x}'\}$.

Since the cylinder $(0, 1] \times (A \cap \partial|B|)$ is homeomorphic to the cone $\text{Cone}(A \cap \partial|B|, \vec{p}) - \{\vec{p}\}$, e.g., by the homeomorphism

$$h_2(t, \vec{x}) := (1 - t)\vec{p} + t\vec{x},$$

we obtain a homeomorphism

$$h_3 := h_2 \circ h_1^{-1} : (A \cap |B|) - \{\vec{p}\} \rightarrow \text{Cone}(A \cap \partial|B|, \vec{p}) - \{\vec{p}\}.$$

The homeomorphism h_3 can easily be extended to the point \vec{p} , resulting in the desired homeomorphism h . \square

We remark that this theorem was already known for semi-algebraic sets in \mathbf{R}^2 , but the proof technique used there was specific for two dimensions [49].

Theorem 4.4.2. *There exists an FO+POLY expressible cone radius query.*

Proof. As is clear from the proof of Theorem 4.4.1, we can define the following cone radius query Q_{radius} , such that for every polynomial constraint database D over \mathcal{S} ,

$$Q_{\text{radius}}(D) := \{(\vec{p}, r) \in \mathbf{R}^n \times \mathbf{R} \mid \vec{p} \in \text{cl}(S^D) \text{ and } r \in (0, \varepsilon)\},$$

where ε is such that (4.4.1) in the proof of Theorem 4.4.1 is satisfied. Let us express this query in FO+POLY.

We define the critical point query Q_{crit} , such that for every polynomial constraint database D over \mathcal{S} ,

$$\begin{aligned} Q_{\text{crit}}(D) := \{(\vec{p}, \vec{x}) \in \mathbf{R}^n \times \mathbf{R}^n \mid & \vec{p} \in \text{cl}(S^D) \\ & \text{and } \vec{x} \in Q_{k\text{-Whitney}}(D) \text{ for a some } k \\ & \text{and } \vec{x} \text{ is a critical point of } f_i|_{Q_{k\text{-Whitney}}(D)} \text{ for a certain } i\}. \end{aligned}$$

The critical point query Q_{crit} can be easily expressed in FO+POLY. Indeed, the consider the following formula $\text{crit}(\vec{p}, \vec{x})$:

$$\begin{aligned} & \text{closure}(\vec{p}) \\ & \wedge \bigvee_{k=0}^{n+1} (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \rightarrow (p_1 - x_1)v_1 = 0)) \vee \dots \\ & \quad \vee (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \rightarrow (p_n - x_n)v_n = 0)) \\ & \quad \vee (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \\ & \quad \quad \quad \rightarrow (p_1 - x_1)v_1 + (p_2 - x_2)v_2 = 0)) \vee \dots \\ & \quad \vee (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \\ & \quad \quad \quad \rightarrow (p_{n-1} - x_{n-1})v_{n-1} + (p_n - x_n)v_n = 0)) \\ & \quad \vee (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \\ & \quad \quad \quad \rightarrow (p_1 - x_1)v_1 + (p_2 - x_2)v_2 + (p_3 - x_3)v_3 = 0)) \vee \dots \\ & \quad \vee (\forall \vec{v}(\text{tangent}(k\text{-Whitney})(\vec{x}, \vec{v}) \\ & \quad \quad \quad \rightarrow (p_1 - x_1)v_1 + \dots + (p_n - x_n)v_n = 0)). \end{aligned}$$

This formula expresses Q_{crit} correctly by Claim 4.4.2. Here, closure denotes the FO+POLY formula expressing the query which returns the closure of the polynomial constraint database, $k\text{-Whitney}$ denotes an FO+POLY formula expressing $Q_{k\text{-Whitney}}$ for $k = 0, \dots, n$ (Lemma 4.2.1), and tangent is an FO+POLY formula expressing Q_{tangent} (Lemma 4.1.1).

The query which returns the set of critical values is expressible in FO+POLY by the formula

$$\text{val}(\vec{p}, r) \equiv \exists \vec{x} (\text{crit}(\vec{p}, \vec{x}) \wedge (f_1(\vec{x}) \vee \dots \vee f_\ell(\vec{x}))).$$

We therefore conclude that the query expressed in FO+POLY as

$$\text{radius}(\vec{p}, r) \equiv \forall r' (\text{val}(\vec{p}, r') \rightarrow r < r'),$$

is a cone radius query, as desired. \square

In the result of the cone radius query $\text{radius}(D)$, each point $\vec{p} \in \text{cl}(S^D)$ is associated with a continuum of cone radii of S^D in \vec{p} . However, by definable choice [71], for each point \vec{p} , we can select a unique representant $r_{\vec{p}}$ in $\{r \mid (\vec{p}, r) \in \text{radius}(D)\}$. Hence, there exists an FO+POLY formula uniqueradius over \mathcal{S} , such that for each point $\vec{p} \in \text{cl}(S^D)$, there exists a unique cone radius r , such that the pair (\vec{p}, r) is in $\text{uniqueradius}(D)$.

We define the following polynomial constraint formula:

$$\gamma_{\text{cone}, A} : \text{cl}(A) \rightarrow \mathbf{R} : \vec{p} \mapsto r_{\vec{p}}, \quad (4.4.2)$$

such that $(\vec{p}, r_{\vec{p}}) \in \text{uniqueradius}(D)$, with D a database over \mathcal{S} such that $S^D = A$.

4.5 The Uniform Cone Radius Decomposition

Although every point of a semi-algebraic set has a cone radius which is strictly greater than zero (Theorem 4.4.1), we are now interested in finding a uniform cone radius for a semi-algebraic set. We define a *uniform cone radius* of a semi-algebraic set $A \subseteq \mathbf{R}^n$ as a real number $\varepsilon_A > 0$ such that ε_A is a cone radius of A in all points of A . For any $X \subseteq A \subseteq \mathbf{R}^n$, we define a uniform cone radius of X with respect to A , as a real number $\varepsilon > 0$ such that ε is a cone radius of A in all points of X .

A first observation is that a uniform cone radius of a semi-algebraic set does not always exist.

Example 4.5.1. Consider the set shown in Figure 4.10. We have drawn the maximal cone radius around the points $\vec{p}_1, \vec{p}_2, \vec{p}_3, \vec{p}_4$, and \vec{p}_5 . It is clear that the closer these points are to the point \vec{p} , the smaller their maximal cone radius. Because we can make the maximal cone radius arbitrarily small by taking points very close to \vec{p} , we may conclude that the set shown in this figure has no uniform cone radius. \square

Let A be a semi-algebraic set in \mathbf{R}^n . Let U_0, \dots, U_m be a finite set of pairwise disjoint semi-algebraic sets in \mathbf{R}^n , which satisfy the following conditions:

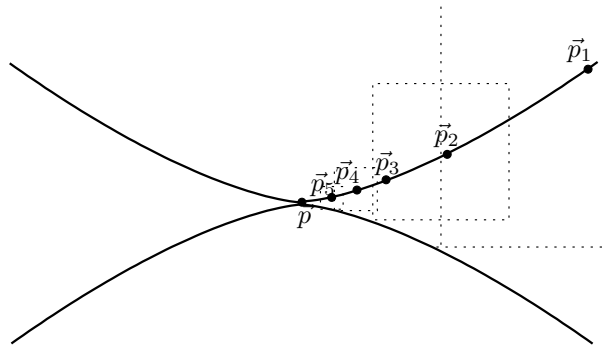


Figure 4.10: Example of a semi-algebraic set which does not have a uniform cone radius.

- The sets U_i , for $i = 0, \dots, m$ form a decomposition of $\text{cl}(A)$, i.e.,

$$\text{cl}(A) = U_0 \cup \dots \cup U_m.$$

- For any m -tuple $(\varepsilon_0, \dots, \varepsilon_m)$ of positive real numbers, and for $i = 0, \dots, m$, the sets

$$U_i - \bigcup_{j=i+1}^m U_j^{\varepsilon_j} \quad (4.5.1)$$

have a uniform cone radius with respect to A if they are nonempty.³

We say that the sets U_0, \dots, U_m form a *uniform cone radius* of A . We now construct a uniform cone radius of A . For any closed subset $X \subset \text{cl}(A)$, we define

$$\Gamma_{\text{nc}}(X) := \{\vec{p} \in X \mid \gamma_{\text{cone}, A} \upharpoonright_X \text{ is not continuous in } \vec{p}\}. \quad (4.5.2)$$

Let $\Delta_0 := \text{cl}(A)$, and let $\Delta_{i+1} := \text{cl}(\Gamma_{\text{nc}}(\Delta_i)) \cap \Delta_i$. We define inductively for $k = 0, 1, \dots, m$, the sets

$$C_k := \Delta_k - \Delta_{k+1}. \quad (4.5.3)$$

Lemma 4.5.1. *For each semi-algebraic set $X \subset \text{cl}(A)$ in \mathbf{R}^n , the set $\Gamma_{\text{nc}}(X)$ is a semi-algebraic set in \mathbf{R}^n and*

$$\dim(\Gamma_{\text{nc}}(X)) < \dim X.$$

³We define the ε -neighborhood of a semi-algebraic set $A \subseteq \mathbf{R}^n$ as

$$A^\varepsilon := \{\vec{x} \in \mathbf{R}^n \mid (\exists \vec{y}) (\vec{y} \in A \wedge \|\vec{x} - \vec{y}\| < \varepsilon)\}.$$

Proof. The set

$$\Gamma_{\text{nc}}(X) = \{\vec{p} \in \mathbf{R}^n \mid (\exists \varepsilon > 0)(\forall \delta > 0)(\exists \vec{p}' \in \mathbf{R}^n)(\vec{p}' \in X \cap B^n(\vec{p}, \delta) \wedge |\gamma_{\text{cone}, A}(\vec{p}) - \gamma_{\text{cone}, A}(\vec{p}')| > \varepsilon)\},$$

is clearly semi-algebraic. This proves the first assertion.

We prove the second assertion by contradiction. Let $d = \dim X$ and suppose that $\dim(\Gamma_{\text{nc}}(X)) = d$, then there exists a semi-algebraic cell $V \subseteq \Gamma_{\text{nc}}(X)$ of dimension d . By the Cell Decomposition Theorem of semi-algebraic sets [71, Theorem 2.11] there exists a semi-algebraic cell decomposition of V into a finite number of semi-algebraic cells,

$$V = V_1 \cup \cdots \cup V_k \cup V_{k+1} \cup \cdots \cup V_\ell,$$

with $\dim(V_i) = d$ for $i = 1, \dots, k$ and $\dim(V_j) < d$ for $j = k+1, \dots, \ell$, such that

$$\gamma_{\text{cone}, A}|_{V_i} \text{ is continuous for every } i = 1, \dots, \ell. \quad (4.5.4)$$

Since $V_i \subset V$ has dimension d for $i = 1, \dots, k$, V_i is open in V , and V_i is also open in X for $i = 1, \dots, k$. From (4.5.4) we deduce that each V_i for $i = 1, \dots, k$ is included in $X - \Gamma_{\text{nc}}(X)$ which is impossible since $V \subseteq \Gamma_{\text{nc}}(X)$. Hence, $\dim(\Gamma_{\text{nc}}(X)) < d$. \square

An immediate consequence of this lemma is that there are at most $n+1$ nonempty sets C_i (or, $m \leq n$).

We now prove that for any tuple $(\varepsilon_0, \dots, \varepsilon_m)$ of positive real numbers, the sets

$$C_i - \bigcup_{j=i+1}^m C_j^{\varepsilon_j}, \quad \text{for } i = 0, 1, \dots, m,$$

have a uniform cone radius if they are nonempty. Since $C_m = \Delta_m$ is closed, $\gamma_{\text{cone}, A}(C_m)$ is also closed and therefore has a minimum which is positive. Hence, C_m has a uniform cone radius. For $i > 0$, we have the inclusion

$$C_i - \bigcup_{j=i+1}^m C_j^{\varepsilon_j} \subseteq \Delta_i - \Delta_{i+1}^\eta, \quad (4.5.5)$$

where $\eta < \min\{\varepsilon_0, \dots, \varepsilon_m\}$. Let $Z = \Delta_i - \Delta_{i+1}^\eta$. Z is closed and $\gamma_{\text{cone}, A}|_Z$ is continuous. Hence, $\gamma_{\text{cone}, A}(Z)$ is closed in \mathbf{R} , and has a minimum which is strict positive. We may conclude that Z has a uniform cone radius, and by (4.5.5) so has the $C_i - \bigcup_{j=i+1}^m C_j^{\varepsilon_j}$. So, C_0, \dots, C_m is a uniform cone radius decomposition of $\text{cl}(A)$.

In Figure 4.11, we have shown the uniform cone radius decomposition of Example 4.5.1.

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. We define the $n+1$ queries $Q_{k\text{-uniform}}$, such that for every polynomial constraint database over \mathcal{S} ,

$$Q_{k\text{-uniform}}(D) := C_k,$$

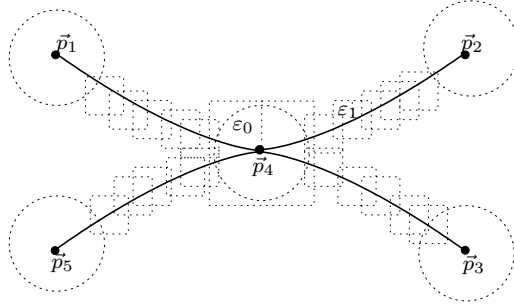


Figure 4.11: The points \vec{p}_1 , \vec{p}_2 , \vec{p}_3 , \vec{p}_4 , and \vec{p}_5 form the part C_0 which has ε_1 as uniform cone radius. As can be seen, the set $C_1 = A - C_0^{\varepsilon_0}$ has a uniform cone radius ε_1 .

for $k = 0, 1, \dots, n$, with C_0, \dots, C_n (with $C_{m+1} = \dots = C_n = \emptyset$) being the uniform cone radius decomposition of S^D .

Because the graph of the mapping $\gamma_{\text{cone}, S^D}$ equals $\text{uniqueradius}(D)$, and by Theorem 4.4.2 uniqueradius is a formula in FO+POLY, the following lemma is immediate.

Lemma 4.5.2. *The queries $Q_{k\text{-uniform}}$, $k = 0, 1, \dots, n$ are expressible in FO+POLY.* \square

4.6 Box Collections

We need one more ingredient before we can start explaining the linearization algorithm. Until now every geometric property we discussed in this chapter was first-order expressible. In this section, we will construct a collection of boxes covering a set in \mathbf{R}^n . We will see that this covering cannot be computed in FO+POLY, but we show how to compute it in FO+POLY+TC.

We define a n -dimensional box collection \mathcal{B} in \mathbf{R}^n as a finite set of n -dimensional boxes satisfying an intersection condition: Let B_1 and B_2 be two arbitrary boxes in \mathcal{B} . Then, if $|B_1|$ and $|B_2|$ intersect, the intersection is included in their boundaries $\partial|B_1|$ and $\partial|B_2|$. By the *geometric realization* $|\mathcal{B}|$ of \mathcal{B} , we mean the union of the geometric realizations of all boxes in \mathcal{B} .

Let D be a set of n -dimensional boxes, which does not necessarily satisfy the above intersection condition. In the following, we show how to split in FO+POLY the boxes in D into smaller boxes, such that the collection of these smaller boxes is a box collection. We call this the *box collection of D* , and denote it with \mathcal{D} . By construction, the geometric realization of each box in D is the union of the geometric realizations of certain boxes of \mathcal{D} .

We first give an example of the construction and then present the general construction more formally.

Example 4.6.1. Fix the dimension $n = 2$, and consider the set D consisting of

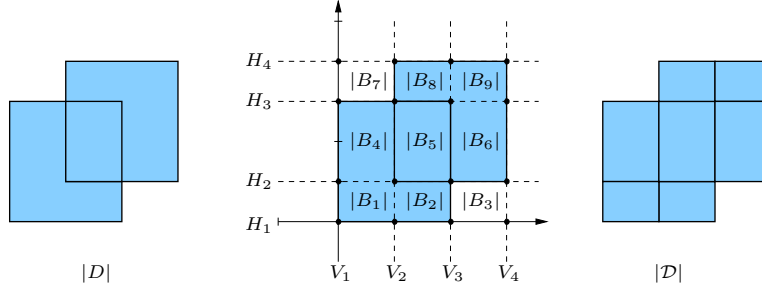


Figure 4.12: A two-dimensional example of the construction of a box collection for two boxes in the \mathbf{R}^2 .

two boxes $(0, 2, 0, 3)$ and $(1, 3, 1, 4)$. The geometric realization $|D|$ of D is depicted in Figure 4.12. In this figure, two sets of lines $\mathcal{H}_{D,x} = \{H_1, H_2, H_3, H_4\}$, and $\mathcal{H}_{D,y} = \{V_1, V_2, V_3, V_4\}$, are drawn. Denote the intersection $\bigcup \mathcal{H}_{D,x} \cap \bigcup \mathcal{H}_{D,y}$ by I . In this example, I consists of 16 points $\{\vec{p}_1, \dots, \vec{p}_{16}\}$. From these points we construct the set \mathcal{P} which contains the 9 two-dimensional boxes denoted by B_i , $i = 1, \dots, 9$. The geometric realizations of these boxes are shown in the figure. As can be seen, these boxes intersect only at their boundaries, and hence form a two-dimensional box collection. Finally, we define the box collection \mathcal{D} of D as the boxes included in $|D|$, i.e., $\mathcal{D} = \{B_1, B_2, B_4, B_6, B_8, B_9\}$. \square

In general, we define n sets of $(n-1)$ -dimensional hyperplanes

$$\mathcal{H}_{D,i} := \{(x_1, \dots, x_n) \in \mathbf{R}^n \mid \exists (a_i, b_1, \dots, a_n, b_n) \in D \wedge (x_i = a_i \vee x_i = b_i)\},$$

for $i = 1, \dots, n$. Let $I \subset \mathbf{R}^n$ be the finite set of points

$$\bigcup \mathcal{H}_{D,1} \cap \dots \cap \bigcup \mathcal{H}_{D,n}.$$

Next, we construct a n -dimensional box collection, which we denote by \mathcal{P} , such that the geometric realization of each box in D is the union of the geometric realizations of boxes in \mathcal{P} . More specifically,

$$\begin{aligned} \mathcal{P} := \{ & (a_1, b_1, \dots, a_n, b_n) \in \mathbf{R}^{2n} \mid \exists \vec{p}_1, \exists \vec{q}_1, \dots, \exists \vec{p}_n, \exists \vec{q}_n \in I \\ & \bigwedge_{i=1}^n (a_i = (\vec{p}_i)_i \wedge b_i = (\vec{q}_i)_i \wedge a_i < b_i) \\ & \wedge (\forall \vec{r} \in I \bigwedge_{i=1}^n \neg (a_i < (\vec{r})_i < b_i))\}. \end{aligned}$$

Finally, we define \mathcal{D} as those n -dimensional boxes B in \mathcal{P} such that $|B|$ is included in the geometric realization of any of the boxes in D . By construction, \mathcal{D} is a box collection, and the geometric realization of any box in D is the union of the geometric

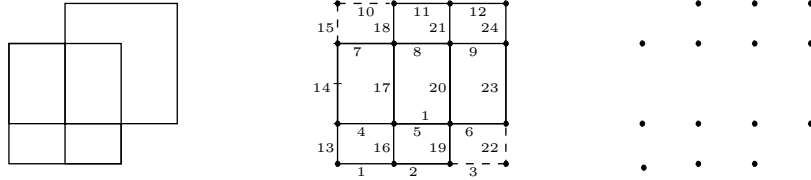


Figure 4.13: The set $|\mathcal{D}| - |\mathcal{D}|_2$ (left). The one-dimensional box collection $\mathcal{P}_x \cup \mathcal{P}_y$ where the line segment L_i is labeled with the number i (center). The set $|\mathcal{D}|_0$ (right).

realizations of certain boxes in \mathcal{D} . The construction of \mathcal{D} for a given D , can be expressed in FO+POLY, as is clear from the above expressions for $\mathcal{H}_{D,i}$ and \mathcal{P} .

This construction can also be applied to the union of box collections, $D \cup D'$. We will denote the resulting box collection by $\mathcal{D} \sqcup \mathcal{D}'$.

Example 4.6.2. (see Figure 4.12 and Figure 4.13). Let us return to the previous example. Let $|\mathcal{D}|_2$ be the set in \mathbf{R}^2 defined by $\bigcup_{i \in \{1,2,4,5,6,8,9\}} \text{int}(|B_i|)$. Consider the set $|\mathcal{D}| - |\mathcal{D}|_2$ and define \mathcal{P}_x to be the set of horizontal line segments L_i , with $i = 1, \dots, 12$, and let \mathcal{P}_y be the set of vertical line segments L_i , with $i = 13, \dots, 24$. The line segments L_i can easily be defined from the points in I and form a one-dimensional box collection. We define \mathcal{D}^1 to be the box collection consisting of boxes in $\mathcal{P}_x \cup \mathcal{P}_y$, which are contained in $|\mathcal{D}|$. Next, define $|\mathcal{D}|_1$ to be the set $\bigcup_{i \in \{1, \dots, 24\} - \{3, 10, 12, 15\}} \text{int}(|L_i|)$. Here, when taking the interior, we regard each $|L_i|$ as a space on itself, so the result is an open line segment without the endpoints (as opposed to the empty set when we would regard each $|L_i|$ as a set in \mathbf{R}^2). Now, $|\mathcal{D}| - |\mathcal{D}|_2 - |\mathcal{D}|_1$ is a subset of I , which we denote by $|\mathcal{D}|_0$. Hence, we have obtained a decomposition of $|\mathcal{D}|$.

This decomposition is important for two reasons. First, the geometric realization of each box of D is the disjoint union of the interiors of the geometric realizations of certain boxes in \mathcal{D}^2 , \mathcal{D}^1 , and \mathcal{D}^0 . Secondly, the interiors of boxes in \mathcal{D} are open subsets of $\text{Reg}_2(|\mathcal{D}|)$, the interiors of boxes in \mathcal{D}^1 are open subsets of $\text{Reg}_1(|\mathcal{D}| - |\mathcal{D}|_2)$, and finally $|\mathcal{D}|_0$ equals $\text{Reg}_0(|\mathcal{D}| - |\mathcal{D}|_2 - |\mathcal{D}|_1)$. \square

In general, the construction of this decomposition goes as follows. For $k = 0, 1, \dots, n$ and any combination of k different elements i_1, \dots, i_k in $\{1, \dots, n\}$, we

define the following set of $n - k$ -dimensional boxes in \mathbf{R}^n :

$$\begin{aligned} \mathcal{P}_{(i_1, \dots, i_k)} := \{ & (a_1, b_1, \dots, a_n, b_n) \in \mathbf{R}^{2n} \mid \exists \vec{p}_1, \exists \vec{q}_1, \dots, \exists \vec{p}_n, \exists \vec{q}_n \in I \\ & \left(\bigwedge_{i \in \{1, \dots, n\} - \{i_1, \dots, i_k\}} (a_i = (\vec{p}_i)_i \wedge (b_i = (\vec{q}_i)_i) \wedge a_i < b_i) \right. \\ & \wedge \forall \vec{r} \in I \bigwedge_{i=1}^n \neg (a_i < (\vec{r})_i < b_i) \\ & \left. \wedge \bigwedge_{i \in \{i_1, \dots, i_k\}} (a_i = (\vec{p}_i)_i \wedge (b_i = (\vec{q}_i)_i) \wedge a_i = b_i) \right\}. \end{aligned} \quad (4.6.1)$$

Note that $\mathcal{P}_{(1, \dots, n)} = I$, and $\mathcal{P}_\emptyset = \mathcal{P}$. It is clear that these sets are expressible in FO+POLY. We also define for $k = 0, 1, \dots, n$ and any combination of k different elements i_1, \dots, i_k in $\{1, \dots, n\}$, the following $n - k$ -dimensional box collection in \mathbf{R}^n :

$$\begin{aligned} \mathcal{D}_{(i_1, \dots, i_k)} := \{ & (a_1, b_1, \dots, a_n, b_n) \in \mathcal{P}_{(i_1, \dots, i_k)} \mid \wedge \exists (a'_1, b'_1, \dots, a'_n, b'_n) \in \mathcal{D} \\ & \wedge \bigwedge_{i=1}^n (a'_i \leq a_i \wedge b_i \leq b'_i) \}. \end{aligned}$$

We then define

$$\mathcal{D}^{n-k} := \bigcup_{(i_1, \dots, i_k)} \mathcal{D}_{(i_1, \dots, i_k)}.$$

Finally, for $k = 0, 1, \dots, n$, we define $|\mathcal{D}|_{n-k}$ as the union of the interiors of the geometric realizations of boxes in \mathcal{D}^{n-k} . Here, when taking the interior, we regard each geometric realization of a box as a space on itself, so the result is an open box. By construction, we have the following properties:

1.

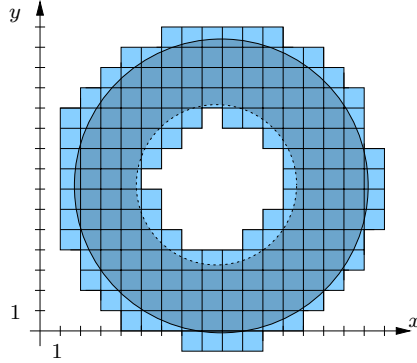
$$|\mathcal{D}| = |\mathcal{D}|_n \cup \dots \cup |\mathcal{D}|_0; \quad (4.6.2)$$

2. each geometric realization of a box in \mathcal{D} is the union of the geometric realizations of boxes in $|\mathcal{D}|_k$, for $k = 0, 1, \dots, n$; and

3. the geometric realizations of boxes in $|\mathcal{D}|_k$ are open subsets of $\text{Reg}_k(|\mathcal{D}| - |\mathcal{D}|_n - \dots - |\mathcal{D}|_{k+1})$.

4.7 Expressing the Box Covering Query

Let $\delta > 0$ be a real number. We define the n -dimensional standard grid of size δ as the n -dimensional box collection δ -grid consisting of all boxes of the form $(k_1\delta, (k_1 + 1)\delta, \dots, k_n\delta, (k_n + 1)\delta)$, where $k_1, k_2, \dots, k_n \in \mathbf{Z}$. We define the *box covering of size δ* of a semi-algebraic set A , denoted with $\delta\text{-cover}(A)$, as those boxes in δ -grid, which intersect the closure of A (see Figure 4.14). Let $\mathcal{S} = \{S\}$, with S an n -ary relation

Figure 4.14: The δ -cover of a semi-open annulus for $\delta = 1$.

name. We define for each $\delta > 0$, the *box covering query* $Q_{\delta\text{-cover}}$, such that for every constraint database D over \mathcal{S} ,

$$Q_{\delta\text{-cover}} := \delta\text{-cover}(S^D).$$

Proposition 4.7.1. *Let $\delta > 0$. The query $Q_{\delta\text{-cover}}$ is not expressible in FO+POLY.*

Proof. Let $\mathcal{S} = \{S\}$, with S a binary relation name. We consider the following FO+POLY formula over \mathcal{S} : a formula `circle` such that for any database D over \mathcal{S} , either `circle`(D) is the circle through the points of S^D , if S^D consists of three non-collinear points, or `circle`(D) = S^D .

Assume that the query $Q_{\delta\text{-cover}}$ is expressible in FO+POLY. Let $\delta\text{-cover}$ be the formula which expresses $Q_{\delta\text{-cover}}$. Then the formula $\varphi \equiv \delta\text{-cover}(\text{circle})$ is also expressible in FO+POLY. However, the number of 4-tuples in $\varphi(D)$ can be made arbitrarily large by choosing D to be a database over \mathcal{S} , such that S^D consists of three points far enough apart. This contradicts the Dichotomy Theorem of Benedikt and Libkin [6], which guarantees the existence of a polynomial p_φ such that $|\varphi(D)| < p_\varphi(|S^D|) = p_\varphi(3)$ in case $|\varphi(D)|$ is finite. \square

However, in FO+POLY+TC we can express the box covering query:

Proposition 4.7.2. *For each $\delta > 0$, the query $Q_{\delta\text{-cover}}$ is expressible in FO+POLY+TC when restricted to bounded input databases.*

Proof. Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. Let D be a database over \mathcal{S} such that S^D is bounded. Let `boundingbox`(x) be a formula over \mathcal{S} with the property that `boundingbox`(D) = $\{M\}$, with M the minimal real number such that $\text{cl}(S^D) \subseteq [-M, M]^n$. It is clear that `boundingbox` \in FO+POLY (see Example 2.1.5). Let

$$\begin{aligned} \text{grid}(u) \equiv & [\text{TC}_{x;x'} \exists M (\text{boundingbox}(M) \wedge x \geq 0 \\ & \wedge x' = x + \delta \wedge x' \leq M)](0, u) \vee u = 0. \end{aligned}$$

Let

$$\delta\text{-cover}(u_1, v_1, \dots, u_n, v_n) \equiv \bigwedge_{i=1}^n (v_i = u_i + \delta \wedge \mathbf{grid}(u_i))$$

$$\wedge \exists \vec{x}(\text{cl}(S)(\vec{x}) \wedge \bigwedge_{i=1}^n u_i < x_i < v_i).$$

Then $Q_{\delta\text{-cover}}(D) = \delta\text{-cover}(D)$ for any database D over \mathcal{S} such that S^D is bounded. \square

5

Linearization and Approximation

In this chapter, we give a construction of both an algebraic linearization and an ε -approximation of semi-algebraic sets which are implementable in FO+POLY+TC. This implementation is based on the construction of a box collection satisfying some special properties. More specifically, it is shown in Section 5.1 how to construct such a box collection. In Section 5.2 we develop an algorithm `LINEARIZE-IN- n -DIMENSIONS`, which uses this special box collection to build the algebraic linearizations and the ε -approximations of bounded semi-algebraic sets. In the same section, we prove the correctness of the algorithm `LINEARIZE-IN- n -DIMENSIONS`. We also show how to extend this algorithm such that it also builds algebraic linearizations of possibly unbounded semi-algebraic sets. Finally, in Section 5.3 we show that after some minor changes, the algorithm `LINEARIZE-IN- n -DIMENSIONS` can be used to build rational linearizations and ε -approximations of semi-algebraic sets.

5.1 Construction of a Special Box Collection

Let \mathcal{B} be an n -dimensional box collection in \mathbf{R}^n , and let $\mathcal{X} = \{X_1, \dots, X_k\}$ be a finite set of pairwise disjoint semi-algebraic sets in \mathbf{R}^n . We now define when \mathcal{B} and \mathcal{X} are in general position. We are going to decompose $|\mathcal{B}|$ and \mathcal{X} into a finite number of regular sets, and then define “being in general position”, in terms of these decompositions.

In (4.6.2), we defined a decomposition of a box collection into regular sets. Applied to $|\mathcal{B}|$, this results in the decomposition $|\mathcal{B}|_n, \dots, |\mathcal{B}|_0$, where $|\mathcal{B}|_i$ is a union of interiors of i -dimensional boxes in \mathbf{R}^n . Similarly, in (4.1.2), we defined a decomposition of semi-

algebraic sets into regular sets. Applied to each X_i , for $i = 1, \dots, k$, this results in the decompositions $R_{i,n}, \dots, R_{i,0}$, where $R_{i,j}$ is a j -dimensional regular set in \mathbf{R}^n .

We say that \mathcal{B} and \mathcal{X} are in general position if and only if $\{|\mathcal{B}|_n, \dots, |\mathcal{B}|_0\}$ and $\{R_{1,n}, \dots, R_{1,0}, \dots, R_{k,n}, \dots, R_{k,0}\}$ are in general position.

5.1.1 Base Case: n -dimensions

Let A be a bounded semi-algebraic set in \mathbf{R}^n .

Statement of the requirements Let $\mathcal{C} = \{C_0, \dots, C_k\}$ be a finite set consisting of the parts of the uniform cone radius decomposition of A , as defined in Section 4.5. Let \mathcal{B} be a fixed n -dimensional box collection in \mathbf{R}^n , which is in general position with \mathcal{U} .

We will construct a box collection \mathcal{R} and a positive real number δ , such that the following properties are satisfied:

- (i) $\text{cl}(C_0 \cup \dots \cup C_k)^\delta \subseteq \text{int}(|R|)$;
- (ii) for all $\tau \in \mathbf{R}^n$ of norm less than δ , $(\mathcal{R} + \tau) \sqcup \mathcal{B} \pitchfork \{C_0, \dots, C_k\}$; and
- (iii) for all $\tau \in \mathbf{R}^n$ of norm less than δ , and for each n -dimensional box $B \in (\mathcal{R} + \tau) \sqcup \mathcal{B}$, $\text{int}(|B|) \cap (C_0 \cup \dots \cup C_k) \neq \emptyset$, and for any point $\vec{p} \in \text{int}(|B|) \cap (C_0 \cup \dots \cup C_k)$, $\gamma_{\text{cone}}(\vec{p}) > \text{diam}(B)$.

Construction We construct \mathcal{R} inductively on the number of parts in \mathcal{C} . For the base case, i.e., when A is the empty set, we let \mathcal{R} be the empty box collection. The properties (i),(ii), and (iii) are then trivially satisfied.

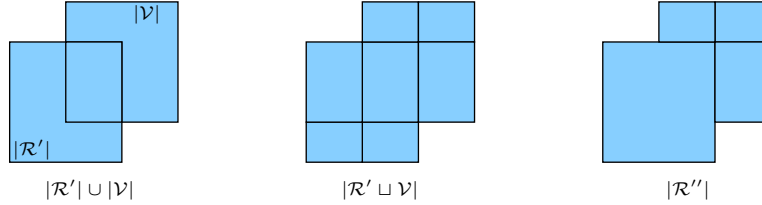
Next, suppose that \mathcal{C} consists of $k + 1$ parts. By induction, there exists a n -dimensional box collection \mathcal{R}' and a positive real number δ' , such that

- (i)' $\text{cl}(C_1 \cup \dots \cup C_k)^{\delta'} \subseteq \text{int}(|R'|)$;
- (ii)' for all $\tau \in \mathbf{R}^n$ of norm less than δ' , $(\mathcal{R}' + \tau) \sqcup \mathcal{B} \pitchfork \{C_1, \dots, C_k\}$; and
- (iii)' for all $\tau \in \mathbf{R}^n$ of norm less than δ' , and for each n -dimensional box $B \in (\mathcal{R}' + \tau) \sqcup \mathcal{B}$, $\text{int}(|B|) \cap (C_1 \cup \dots \cup C_k) \neq \emptyset$, and for any point $\vec{p} \in \text{int}(|B|) \cap (C_1 \cup \dots \cup C_k)$, $\gamma_{\text{cone}}(\vec{p}) > \text{diam}(B)$.

First step: extending \mathcal{R}' We extend \mathcal{R}' to a box collection \mathcal{R}'' such that $\text{cl}(C_0 \cup \dots \cup C_k)^{\delta''} \subseteq \text{int}(|\mathcal{R}''|)$ for some $\delta'' > 0$.

All points of C_0 that can become uncovered by translating $|\mathcal{R}'|$ by a vector τ with $\|\tau\| < \frac{\delta'}{3}$ are included in the following set.

$$V := C_0 - (|\mathcal{R}'| - (\partial|\mathcal{R}'|)^{\frac{\delta'}{3}}).$$

Figure 5.1: Extending \mathcal{R}' to \mathcal{R}'' , keeping \mathcal{R}' intact.

By (i)', the minimal distance from any point in $C_1 \cup \dots \cup C_k$ to the boundary $\partial(|\mathcal{R}'|)$ is greater than or equal to δ' . This implies that

$$\text{cl}(C_1 \cup \dots \cup C_k)^{\frac{\delta'}{3}} \subset |\mathcal{R}'| - (\partial|\mathcal{R}'|)^{\frac{\delta'}{3}},$$

and hence, because C_0, \dots, C_k is a uniform cone radius decomposition, there exists a uniform cone radius, ε_V , of A for the set V . Let \mathcal{V} be $\frac{\varepsilon_V}{4\sqrt{n}}$ -cover(V) such that

$$\text{diam}(B) = \frac{\varepsilon_V}{2} < \varepsilon_V \quad (5.1.1)$$

for any box $B \in \mathcal{V}$. The reason why we take this specific box covering is that the box collection, which we are constructing, must satisfy property (iii).

We now take the union $\mathcal{R}' \sqcup \mathcal{V}$. However, since \mathcal{R}' already has the desired properties, we only want to add boxes outside $|\mathcal{R}'|$. This can be done, as illustrated in Figure 5.1, by performing the following steps: First, take the union $\mathcal{R}' \sqcup \mathcal{V}$, which is shown in the middle of Figure 5.1. Next, remove the n -dimensional boxes in $\mathcal{R}' \sqcup \mathcal{V}$, whose geometric realization is included in $|\mathcal{R}'|$, and finally, add the boxes in \mathcal{R}' again. The resulting box collection, \mathcal{R}'' , is shown at the right in Figure 5.1.

We now show that there exists a positive real number δ'' such that (i) holds for \mathcal{R}'' and δ'' .

We partition $C_0 \cup \dots \cup C_k$ into three parts: $C_1 \cup \dots \cup C_k$, V , and $W := C_0 \cap (|\mathcal{R}'| - (\partial|\mathcal{R}'|)^{\frac{\delta'}{3}})$. By the induction hypothesis,

$$\text{cl}(C_1 \cup \dots \cup C_k)^{\frac{\delta'}{3}} \subseteq \text{int}(|\mathcal{R}'|) \subseteq \text{int}(|\mathcal{R}''|). \quad (5.1.2)$$

We shall need the following property.

Claim 5.1.1. *Let X and Y be two sets in \mathbf{R}^n . If X is bounded, then $\text{cl}(X) \subseteq \text{int}(Y)$ implies that there exists a positive real number ε such that $\text{cl}(X)^\varepsilon \subseteq \text{int}(Y)$.*

Proof of Claim. Suppose that for any $m \in \mathbf{N}$, $m > 0$, $\text{cl}(X)^{1/m} \not\subseteq \text{int}(Y)$. Then there exist two sequences of points $(\vec{x}_m) \in \text{cl}(X)$ and $(\vec{y}_m) \in (\mathbf{R}^n - \text{int}(Y))$, such that $\|\vec{x}_m - \vec{y}_m\| < 1/m$. Since $\text{cl}(X)$ is compact, the sequence (\vec{x}_m) has a convergent subsequence (\vec{x}_{m_k}) . Let $\vec{x} \in \text{cl}(X)$ be the limit point of this sequence. Since $\mathbf{R}^n - \text{int}(Y)$ is closed, the corresponding sequence (\vec{y}_{m_k}) has also a limit point $\vec{y} \in (\mathbf{R}^n - \text{int}(Y))$. However, \vec{x} must be equal to \vec{y} , but this is impossible by the assumption that $\text{cl}(X) \cap (\mathbf{R}^n - \text{int}(Y)) = \emptyset$. \square

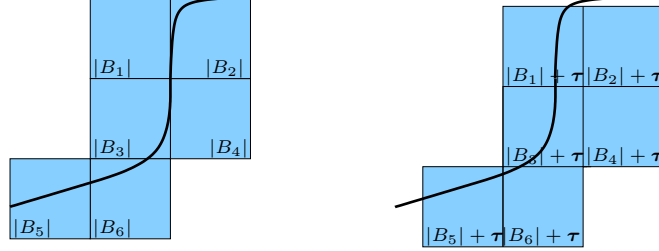


Figure 5.2: A box collection (left) is brought into general position with the thick curve by translating it (right).

By definition of a box covering, $\text{cl}(V) \subseteq \text{int}(|\mathcal{V}|) \subseteq \text{int}(|\mathcal{R}''|)$. Since A is bounded, V is also bounded. By Claim 5.1.1, there exists a positive real number η such that,

$$\text{cl}(V)^\eta \subset \text{int}(|\mathcal{R}''|). \quad (5.1.3)$$

We now prove that Claim 5.1.1 can also be used for W .

Claim 5.1.2. $\text{cl}(W) \subseteq \text{int}(|\mathcal{R}'|)$

Proof of Claim. Suppose that there exists a point $\vec{p} \in \text{cl}(W)$ such that $\vec{p} \notin \text{int}(|\mathcal{R}'|)$. Let $m \in \mathbf{N}$, $m > 0$, and let (\vec{p}_m) be a sequence of points in W , such that $\|\vec{p} - \vec{p}_m\| < 1/m$. By the definition of W , for all points in $\vec{r} \in \partial|\mathcal{R}'|$, $\|\vec{r} - \vec{p}_m\| \geq \frac{\delta'}{3}$, for all $m \in \mathbf{N}$.

Now, every line segment $\lambda\vec{p}_m + (1 - \lambda)\vec{p}$, for $0 \leq \lambda \leq 1$, intersects $\partial|\mathcal{R}'|$ in a point \vec{r}_m . However, since $\|\vec{p}_m - \vec{p}\| < 1/m$, also $\|\vec{p}_m - \vec{r}_m\| < 1/m$. So, we obtain a contradiction for m large enough such that $\frac{1}{m} < \frac{\delta'}{3}$. \square

Hence, by Claim 5.1.1 and Claim 5.1.2 there exists a positive real number ζ , such that

$$W^\zeta \subseteq \text{int}(|\mathcal{R}'|) \subseteq \text{int}(|\mathcal{R}''|). \quad (5.1.4)$$

From the inclusions (5.1.2), (5.1.3), and (5.1.4), it follows that property (i) is satisfied for \mathcal{R}'' and δ'' , with $\delta'' = \min\{\frac{\delta'}{3}, \eta, \zeta\}$.

Second step: translating \mathcal{R}'' The box collection \mathcal{R}'' already satisfies property (i) for δ'' . However, properties (ii) and (iii) are not necessarily satisfied. Indeed, at the left in Figure 5.2, we have drawn a piece of C_0 together with a box collection $\mathcal{R}'' \sqcup \mathcal{B}$. As can easily be verified, property (ii) is violated in $|B_1|, |B_2|, |B_3|$, and $|B_4|$ for $\tau = \vec{0}$. Property (iii) is violated only in $|B_1|$ and $|B_4|$, for $\tau = \vec{0}$, because C_0 only intersects the boundaries of these boxes. As can be seen at the right in Figure 5.2, after translating the 6 boxes, we can obtain a box collection which satisfies both properties (ii) and (iii).

Claim 5.1.3. *There exists a translation $\tau \in \mathbf{R}^n$ of norm $\|\tau\| < \delta''$, such that*

$$(\mathcal{R}'' + \tau) \sqcup \mathcal{B} \pitchfork \{C_0, \dots, C_k\}.$$

Proof of Claim. Consider the decomposition of $|(\mathcal{R}'' + \tau) \sqcup \mathcal{B}|$ into the sets $|(\mathcal{R}'' + \tau) \sqcup \mathcal{B}|_i$, for $i = 0, 1, \dots, n$. Recall that $|(\mathcal{R}'' + \tau) \sqcup \mathcal{B}|_i$ is the union of the geometric realizations of boxes in $B \in ((\mathcal{R}'' + \tau) \sqcup \mathcal{B})^i$.

We need to prove that there exists a translation $\tau \in \mathbf{R}^n$, of norm $0 \leq \|\tau\| < \delta''$, such that for any $j = 0, \dots, n$, and for each box $B \in ((\mathcal{R}'' + \tau) \sqcup \mathcal{B})^j$,

$$|B| \pitchfork R_{i,r} \text{ for } i = 0, \dots, k, r = 0, \dots, n - i, \quad (5.1.5)$$

where $R_{i,r} = \text{Reg}_r(C_i)$. Let T be the set $\{\tau \in \mathbf{R}^n \mid 0 \leq \|\tau\| < \delta''\}$. We are going to impose several conditions on T , such that if $\tau \in T$ and τ satisfies these conditions, then (5.1.5) holds for τ .

So, let $B \in ((\mathcal{R}'' + \tau) \sqcup \mathcal{B})^j$ for some $j \in \{0, \dots, n\}$ and consider $\vec{x} \in |B| \cap R_{i,r}$ for some $i = 0, \dots, k$ and $r = 0, \dots, n - i$. We distinguish between the following cases:

1. $i > 0$. By construction of \mathcal{R}'' , $B \in ((\mathcal{R}' + \tau) \sqcup \mathcal{B})^j$ for any $\tau \in \mathbf{R}^n$, of norm less than δ'' . Hence, by the induction hypothesis, $|B| \pitchfork C_i$. In particular, $|B|$ and $R_{i,r}$ are transversal in \vec{x} for all $\tau \in T$.
2. $i = 0$. By definition of the union operator \sqcup , there exists a neighborhood W of \vec{x} such that one of the following three cases holds:

- (a) $|B| \cap W = |B'| \cap W$ for some $B' \in \mathcal{B}^p$ for some p . Note that,

$$\mathbb{T}_{\vec{x}}|B| = \mathbb{T}_{\vec{x}}(|B| \cap W) = \mathbb{T}_{\vec{x}}(|B'| \cap W) = \mathbb{T}_{\vec{x}}|B'|. \quad (5.1.6)$$

By the given that, $\mathcal{B} \pitchfork C_0$, $|B'|$ and $R_{0,r}$ are transversal in \vec{x} for all $\tau \in T$. By (5.1.6), we may conclude that $|B|$ and $R_{0,r}$ are transversal in \vec{x} for all $\tau \in T$.

- (b) $|B| \cap W = |B''| \cap W$ for some $B'' \in (\mathcal{R}'' + \tau)^q$ for some q . Note that,

$$\mathbb{T}_{\vec{x}}|B| = \mathbb{T}_{\vec{x}}(|B| \cap W) = \mathbb{T}_{\vec{x}}(|B''| \cap W) = \mathbb{T}_{\vec{x}}|B''|. \quad (5.1.7)$$

Suppose that

$$(\mathcal{R}'' + \tau) \pitchfork C_0. \quad (\text{T1})$$

Then, $|B''| \pitchfork C_0$ and hence, $|B''|$ and $R_{0,r}$ are transversal in \vec{x} for all $\tau \in T$ such that condition (T1) is satisfied. By (5.1.7), we may conclude that $|B|$ and $R_{0,r}$ are transversal in \vec{x} for all $\tau \in T$ such that condition (T1) is satisfied.

- (c) $|B| \cap W = |B'| \cap |B''| \cap W$ for some $B' \in \mathcal{B}^p$ for some p , and for some $B'' \in (\mathcal{R}'' + \tau)^q$ for some q . Suppose that

$$(|\mathcal{R}''| + \tau) \pitchfork |\mathcal{B}|. \quad (\text{T2})$$

Because the intersection of regular sets in general position is regular, the tangent space $\mathbb{T}_{\vec{x}}(|B'| \cap |B''|)$ exists. Note that,

$$\mathbb{T}_{\vec{x}}|B| = \mathbb{T}_{\vec{x}}(|B| \cap W) = \mathbb{T}_{\vec{x}}(|B'| \cap |B''| \cap W) = \mathbb{T}_{\vec{x}}(|B'| \cap |B''|). \quad (5.1.8)$$

Furthermore, suppose that in $|B'|$,¹

$$(|B'| \cap |B''|) \pitchfork (|B'| \cap R_{0,r}). \quad (\text{T3})$$

In particular, $\text{T}_{\vec{x}}(|B'| \cap |B''|) + \text{T}_{\vec{x}}(|B'| \cap R_{0,r}) = \text{T}_{\vec{x}}|B'|$. By the given that $\mathcal{B} \pitchfork C_0$, $\text{T}_{\vec{x}}|B'| + \text{T}_{\vec{x}}R_{0,r} = \mathbf{R}^n$. Because the tangent space of the intersection of two regular sets in a point in this intersection, is the intersection of the tangent spaces of the regular sets in this point, we have that $\text{T}_{\vec{x}}(|B'| \cap R_{0,r}) \subseteq \text{T}_{\vec{x}}(R_{0,r})$. Hence,

$$\begin{aligned} \text{T}_{\vec{x}}|B| + \text{T}_{\vec{x}}(R_{0,r}) &= \text{T}_{\vec{x}}(|B'| \cap |B''|) + \text{T}_{\vec{x}}(R_{0,r}) \\ &= \text{T}_{\vec{x}}(|B'| \cap |B''|) + \text{T}_{\vec{x}}(|B'| \cap R_{0,r}) + \text{T}_{\vec{x}}(R_{0,r}) \\ &= \text{T}_{\vec{x}}(|B'|) + \text{T}_{\vec{x}}(R_{0,r}) \\ &= \mathbf{R}^n. \end{aligned}$$

Hence, we may conclude that $|B|$ and $R_{0,r}$ are transversal in \vec{x} for all $\tau \in T$ such that conditions (T2) and (T3) are satisfied.

We may conclude that $(\mathcal{R}'' + \tau) \sqcup \mathcal{B} \pitchfork \mathcal{C}$, if $\tau \in T$ and τ is such that for each box $B \in ((\mathcal{R}'' + \tau) \sqcup \mathcal{B})^j$ for $j = 0, \dots, n$, either no extra condition holds, the condition (T1) holds, or both conditions (T2) and (T3) hold. Hence, we obtain a finite number of conditions on the translations in T . By Corollary 4.3.1, the set of translations $\tau \in T$ for which a single transversality condition, like (T1), (T2), and (T3), is not satisfied, has measure zero. Since a finite union of sets of measure zero, also have measure zero, this implies that for almost all translations in T , all conditions can be satisfied simultaneously. This concludes the proof of the claim. \square

Let τ_0 be a translation, as specified in Claim 5.1.3. Define $\mathcal{R} := \mathcal{R}'' + \tau_0$. By stability of transversality, there exists a positive real number τ such that $(\mathcal{R} + \tau) \sqcup \mathcal{B} \pitchfork \mathcal{C}$ for any $\tau \in \mathbf{R}^n$ such that $0 \leq \|\tau\| < \tau$. Let $\delta = \min\{\delta'' - \|\tau_0\|, \tau\}$.

Claim 5.1.4. *The box collection \mathcal{R} and δ are such that property (i), (ii) and (iii) are satisfied.*

Proof of Claim. We first prove that property (i) is satisfied. We know already that this property is satisfied for \mathcal{R}'' and δ'' . It follows that

$$\text{cl}(C_0 \cup \dots \cup C_k) \subseteq \text{int}(|\mathcal{R}''|) + \tau, \quad \text{for all } \tau \in \mathbf{R}^n, \|\tau\| < \delta''. \quad (5.1.9)$$

Indeed, suppose that (5.1.9) is not true. Then there exists a point $\vec{p} \in \text{cl}(C_0 \cup \dots \cup C_k)$ and a vector $\tau_1 \in \mathbf{R}^n$, $\|\tau_1\| < \delta''$, such that $\vec{p} \notin \text{int}(|\mathcal{R}''|) + \tau_1$. Define $\vec{x} = \vec{p} - \tau_1$. Then, $\vec{x} \notin \text{int}(|\mathcal{R}''|)$ and $\vec{x} \in \text{cl}(C_0 \cup \dots \cup C_k)^{\delta''}$. This is impossible since \mathcal{R}'' and δ'' satisfy property (i). As a result, (5.1.9) holds. In particular,

$$\text{cl}(C_0 \cup \dots \cup C_k) \subseteq \text{int}(|\mathcal{R}''|) + \tau_0 + \tau, \quad \text{for all } \tau \in \mathbf{R}^n, \|\tau\| < \delta'' - \|\tau_0\|. \quad (5.1.10)$$

¹If X and Y are two regular subsets of a regular set Z , then X and Y are called transversal in Z , if $\text{T}_{\vec{x}}X + \text{T}_{\vec{x}}Y = \text{T}_{\vec{x}}Z$, for any point in the intersection $X \cap Y$.

Now, suppose that $\text{cl}(C_0 \cup \dots \cup C_k)^\delta \not\subseteq \text{int}(|\mathcal{R}|)$. Then there exists a point $\vec{p} \in \text{cl}(C_0 \cup \dots \cup C_k)^\delta$ such that $\vec{p} \notin \text{int}(|\mathcal{R}|)$. Moreover, there exists a point $\vec{x} \in \text{cl}(C_0 \cup \dots \cup C_k)$ such that $\|\vec{p} - \vec{x}\| < \delta$. Define $\tau_1 = \vec{x} - \vec{p}$. Because $\vec{x} \in \text{cl}(C_0 \cup \dots \cup C_k)$ and $\vec{x} = \vec{p} + \tau_1$ with $\vec{p} \notin \text{int}(|\mathcal{R}|)$,

$$\text{cl}(C_0 \cup \dots \cup C_k) \not\subseteq \text{int}(|\mathcal{R}''|) + \tau_0 + \tau_1 = \text{int}(|\mathcal{R}|) + \tau_1,$$

which contradicts (5.1.10) because $\|\tau_1\| < \delta$. Hence, $\text{cl}(C_0 \cup \dots \cup C_k)^\delta \subset \text{int}(|\mathcal{R}|)$ and property (i) is satisfied for \mathcal{R} and δ .

Property (ii) is trivially satisfied for \mathcal{R} and δ by the definition of δ and τ_0 .

We now prove that property (iii) is satisfied. Let $B \in (\mathcal{R} + \tau) \sqcup \mathcal{B}$. We distinguish between the following two cases:

1. $B \in (\mathcal{R}' + \tau_0 + \tau) \sqcup \mathcal{B}$. Note that $\|\tau_0 + \tau\| < \delta'$. Hence, by the induction hypothesis, $\text{int}(|B|) \cap (C_1 \cup \dots \cup C_k) \neq \emptyset$, and for any point $\vec{p} \in \text{int}(|B|) \cap (C_1 \cup \dots \cup C_k)$, $\gamma_{\text{cone}}(\vec{p}) > \text{diam}(B)$.
2. $B \in ((\mathcal{R}'' - \mathcal{R}') + \tau_0 + \tau) \sqcup \mathcal{B}$. It remains to show that $\text{int}(|B|) \cap C_0 \neq \emptyset$ and for any point $\vec{p} \in \text{int}(|B|) \cap C_0$, $\gamma_{\text{cone}}(\vec{p}) > \text{diam}(B)$.

We first prove that $|B| \cap C_0 \neq \emptyset$ implies that $\text{int}(|B|) \cap C_0 \neq \emptyset$.

So let, $\vec{x} \in |B| \cap C_0$. If $\vec{x} \in \text{int}(|B|)$, we are done. Thus, we assume that $\vec{x} \in \partial|B|$. Then $\vec{x} \in |B'| \cap C_0$ for some $|B'| \in ((\mathcal{R} + \tau) \sqcup \mathcal{B})^p$ and some p . Let $D = (x_1 - \varepsilon, x_1 + \varepsilon, \dots, x_n - \varepsilon, x_n + \varepsilon)$ be an n -dimensional box centered around \vec{x} , with $\varepsilon \in \mathbf{R}$. For ε sufficiently small, $|B'| \cap \text{int}(|D|)$ has the form

$$(x_1 - \varepsilon, x_1 + \varepsilon) \times \dots \times (x_p - \varepsilon, x_p + \varepsilon) \times \{x_{p+1}\} \times \dots \times \{x_n\},$$

or a permutation of this form, which is handled analogously. Hence, $\text{int}(|B|) \cap \text{int}(|D|)$ has the form

$$(x_1 - \varepsilon, x_1 + \varepsilon) \times \dots \times (x_p - \varepsilon, x_p + \varepsilon) \times (x_{p+1}, x_{p+1} + \varepsilon) \times \dots \times (x_n, x_n + \varepsilon),$$

or a permutation of this form which is handled analogously, or even a variant of this form where some of the $n - p$ intervals $(x_i, x_i + \varepsilon)$ are replaced by $(x_i - \varepsilon, x_i)$, which again is handled analogously.

By property (ii),

$$\mathbb{T}_{\vec{x}}|B'| + \mathbb{T}_{\vec{x}}C_0 = \mathbf{R}^n. \quad (5.1.11)$$

Now, any $\vec{v} \in \mathbb{T}_{\vec{x}}|B'|$ is of the form $\vec{v} = (v_1, \dots, v_p, x_{p+1}, \dots, x_n)$, hence, by (5.1.11) there exists a tangent vector $\vec{w} \in \mathbb{T}_{\vec{x}}C_0$ such that $x_{p+1} < w_{p+1}$, $\dots, x_n < w_n$. By definition of the tangent space, if $\|\vec{w} - \vec{x}\|$ is small enough, there exists a point in C_0 arbitrary close to \vec{w} . This point, then also has $n - p$ last coordinates which are strictly greater than the $n - p$ last coordinates of \vec{x} , and hence, is in $\text{int}(|B|) \cap |D|$. Hence, we have found a point in $\text{int}(|B|) \cap C_0$, as desired.

This reasoning actually shows that by property (ii), the set of boxes in $((\mathcal{R}'' - \mathcal{R}') + \tau) \sqcup \mathcal{B}$ which intersect C_0 is independent of τ . Hence, we may assume that $|B| \cap C_0 \neq \emptyset$ for any box in $((\mathcal{R}'' - \mathcal{R}') + \tau) \sqcup \mathcal{B}$ and any $\tau \in \mathbf{R}^n$, $\|\tau\| < \delta$.

It remains to show that the diameter of B is smaller than $\gamma_{\text{cone}}(\vec{p})$, for a point $\vec{p} \in \text{int}(|B|) \cap C_0$. Now, any box in $((\mathcal{R}'' - \mathcal{R}') + \tau_0 + \tau) \sqcup \mathcal{B}$ is included in a box in $\mathcal{V} + \tau_0 + \tau$. By (5.1.1), \mathcal{V} consists of boxes which have a diameter which is strictly smaller than the uniform cone radius of $\text{int}(|B|) \cap C_0$. Hence, $\gamma_{\text{cone}}(\vec{p}) > \text{diam}(|B|)$ for any point in $\text{int}(|B|) \cap C_0$.

As a result, property (iii) is satisfied for \mathcal{R} and δ . \square

5.1.2 Induction: from n dimensions to $n - 1$

From \mathcal{R} we compute the n -dimensional box collections $\mathcal{R}_{(i)}$ for $i = 1, \dots, n$, (see (4.6.1)), and we define the semi-algebraic sets

$$\text{Coord}(\mathcal{R}_{(i)}) = \{a \in \mathbf{R} \mid \exists a_1, \exists b_1, \dots, \exists a_{i-1}, \exists b_{i-1}, \exists a_{i+1}, \exists b_{i+1}, \dots, \exists a_n, \exists b_n \\ (a_1, b_1, \dots, a_{i-1}, b_{i-1}, a, a, a_{i+1}, b_{i+1}, \dots, a_n, b_n) \in \mathcal{R}_{(i)}\},$$

and

$$A_{(i), a_i} := \{(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \in \mathbf{R}^{n-1} \mid \\ (x_1, \dots, x_{i-1}, a_i, x_{i+1}, \dots, x_n) \in |\mathcal{R}_{(i)}| \cap A\}$$

for $i = 1, \dots, n$.

Similarly, we define the $(n - 1)$ -dimensional box collections

$$\mathcal{R}_{(i), a_i} := \{(a_1, b_1, \dots, b_{i-1}, a_{i+1}, \dots, a_n, b_n) \in \mathbf{R}^{2(n-1)} \mid \\ (a_1, b_1, \dots, b_{i-1}, a_i, a_i, a_{i+1}, \dots, a_n, b_n) \in \mathcal{R}_{(i)}\},$$

for $i = 1, \dots, n$.

Since $A = C_0 \cup \dots \cup C_k$, and $C_j = R_{j,0} \cup \dots \cup R_{j,j}$, we have that

$$A_{(i), a_i} = (C_0)_{(i), a_i} \cup \dots \cup (C_k)_{(i), a_i} \quad (5.1.12)$$

$$(C_j)_{(i), a_i} = (R_{j,0})_{(i), a_i} \cup \dots \cup (R_{j,j})_{(i), a_i}. \quad (5.1.13)$$

Claim. *The sets $(C_0)_{(i), a_i}, \dots, (C_k)_{(i), a_i}$ form a uniform cone radius decomposition of $A_{(i), a_i}$.*

Proof. The proof of Theorem 4.4.1 shows that a cone radius ε of A in a point \vec{p} is also a cone radius of $A \cap H$ for \vec{p} , if $\vec{p} \in H$, with H a hyperplane parallel to one of the coordinate planes. Indeed, the critical points of $A \cap H$ are already identified in the computation of ε , and hence condition (4.4.1) is trivially satisfied with respect to $A \cap H$ and \vec{p} . \square

Claim. *The sets $(R_{j,0})_{(i), a_i}, \dots, (R_{j,j})_{(i), a_i}$ form the regular decomposition of $(C_j)_{(i), a_i}$.*

Proof. The sets $(R_{j,r})_{(i),a_i} = R_{j,r} \cap |\mathcal{R}_{(i)}|$. Since $\mathcal{R} \pitchfork R_{j,r}$, and the intersection of two regular sets in general position is regular, $(R_{j,r})_{(i),a_i}$ is indeed regular. \square

Claim.

$$\mathcal{R}_{(i),a_i} \pitchfork \{(C_0)_{(i),a_i}, \dots, (C_k)_{(i),a_i}\}.$$

Proof. Let $B \in (\mathcal{R}_{(i),a_i})^p$ for some p . We need to show that $|B|$ and $(R_{j,r})_{(i),a_i}$ are in general position. Let $\vec{x} \in |B| \cap R_{j,r}$. Now, $(v_1, \dots, v_{n-1}) \in T_{\vec{x}}((R_{j,r})_{(i),a_i})$ if and only if $(v_1, \dots, v_{i-1}, a_i, v_{i+1}, \dots, v_{n-1}) \in T_{\vec{x}}(R_{j,r} \cap |B'|)$ for some $B' \in \mathcal{R}_{(i),a_i}$. Because $T_{\vec{x}}|B| + T_{\vec{x}}(R_{j,r}) = \mathbf{R}^n$, we have that $T_{\vec{x}}(R_{j,r} \cap |B'|) = T_{\vec{x}}R_{j,r} \cap T_{\vec{x}}|B'|$. So, $(v_1, \dots, v_{n-1}) \in T_{\vec{x}}((R_{j,r})_{(i),a_i})$ if and only if $(v_1, \dots, v_{i-1}, a_i, v_{i+1}, \dots, v_{n-1}) \in T_{\vec{x}}(R_{j,r})$. Hence, $T_{\vec{x}}|B| + T_{\vec{x}}((R_{j,r})_{(i),a_i}) = \mathbf{R}^{n-1}$. \square

5.2 The Algorithm

The algorithm that constructs an \mathbf{A} -linear set which is homeomorphic to a given semi-algebraic set, works inductively on the dimension of the surrounding space in which the semi-algebraic set is embedded.

The bounded case The algorithm consists of two parts. The first part is a preprocessing step: One gives a bounded semi-algebraic set A in \mathbf{R}^n , together with the dimension n as input. As output one gets the regular decomposition of each part of the uniform cone radius decomposition of A .

Algorithm PREPROCESS:

Input: A semi-algebraic set A in \mathbf{R}^n .

Output: A finite sequence $R_{0,0}, R_{1,1}, R_{1,0}, \dots, R_{n,n}, R_{n,n-1}, \dots, R_{n,0}$ of semi-algebraic sets in \mathbf{R}^n .

Method:

1. Compute the uniform cone radius decomposition of A :

$$A = C_0 \cup \dots \cup C_k.$$

2. Compute the regular decomposition of C_i , for $i = 0, \dots, k$:

$$C_i = R_{i,0} \cup \dots \cup R_{i,i}.$$

After this we can feed the output of the preprocessing, combined with an empty box collection \mathcal{B} to the linearization algorithm:

Algorithm LINEARIZE-IN- n -DIMENSIONS:

Input: $(\{R_{j,r}\}, \mathcal{B})$, with $\{R_{j,r}\}$ a regular decomposition of a semi-algebraic set A in \mathbf{R}^n , and \mathcal{B} an n -dimensional box collection in \mathbf{R}^n .

Output: An \mathbf{A} -linear set \widehat{A} in \mathbf{R}^n which is homeomorphic to A

Method:

1. Compute the box collection \mathcal{R} constructed in Section 5.1
2. Compute the $(3n+1)$ -ary relation \mathcal{P} consisting of all pairs (B, \vec{p}_B, b) , where B is an n -dimensional box in \mathcal{R} , $\vec{p}_B \in \mathbf{R}^n$ is the center of $|B|$, and either $b = 1$, if $A \cap |B|$ is homeomorphic to $\text{Cone}(A \cap \partial|B|, \vec{p})$ for some $\vec{p} \in \text{int}(|B|)$, or $b = 0$, if $A \cap |B|$ is homeomorphic to $\text{Cone}(A \cap \partial|B|, \vec{p}) - \{\vec{p}\}$ for some $\vec{p} \in \text{int}(|B|)$.
3. In case $n > 0$ do the following:
 - (a) Compute $\mathcal{R}_{(i),a}$ with $a \in \text{Coord}(\mathcal{R}_{(i)})$ and $i \in \{1, \dots, n\}$.
 - (b) Compute $(R_{j,r})_{(i),a} \subset \mathbf{R}^{n-1}$ with $a \in \text{Coord}(\mathcal{R}_{(i)})$ and $i \in \{1, \dots, n\}$.
 - (c) Apply LINEARIZE-IN- $(n-1)$ -DIMENSIONS in parallel to all input pairs $((R_{j,r})_{(i),a}, \mathcal{R}_{(i),a_i})$ with $a \in \text{Coord}(\mathcal{R}_{(i)})$ and $i \in \{1, \dots, n\}$.
4. Initialize \widehat{A} to the union of the results of the calls to LINEARIZE-IN- $(n-1)$ -DIMENSIONS of step 3(c).
5. Output

$$\begin{aligned} \widehat{A} := & \widehat{A} \cup \{ \text{Cone}(\widehat{A} \cap \partial B, \vec{p}_B) \mid (B, \vec{p}_B, b) \in \mathcal{P} \text{ and } b = 1 \} \\ & \cup \{ \text{Cone}(\widehat{A} \cap \partial B, \vec{p}_B) - \{\vec{p}_B\} \mid (B, \vec{p}_B, b) \in \mathcal{P} \text{ and } b = 0 \}. \end{aligned}$$

Theorem 5.2.1. *For each n there exists an FO+POLY+TC formula `linearize` over the schema $\mathcal{S} = \{S\}$, with S an n -ary relation name, such that for any polynomial constraint database D over \mathcal{S} , `linearize`(D) is an algebraic linearization of S^D , if S^D is bounded.*

Proof. The desired FO+POLY+TC formula `linearize` expresses the algorithms PREPROCESS and LINEARIZE-IN- n -DIMENSIONS described above. From Lemma 4.5.2 and Lemma 4.1.2, it follows that the algorithm PREPROCESS is FO+POLY-expressible. What concerns the algorithm LINEARIZE-IN- n -DIMENSIONS, from Lemma 4.3.1, from the constructions in Section 4.6 and from the construction of the special box collection in Section 5.1, it follows that these are all expressible in FO+POLY+TC.

We still need to show that \widehat{A} is indeed a linearization of A . The linearity of \widehat{A} is immediate, so we focus on the existence of a homeomorphism $h : \mathbf{R}^n \rightarrow \mathbf{R}^n$ which maps A to \widehat{A} .

Let $\mathcal{R}^{\leq k}$ be the union of all boxes in $\mathcal{R}^k, \dots, \mathcal{R}^0$. We shall construct homeomorphisms $h_k : |\mathcal{R}^{\leq k}| \rightarrow |\mathcal{R}^{\leq k}|$, such that

- $h_k(A \cap |\mathcal{R}^{\leq k}|) = \widehat{A} \cap |\mathcal{R}^{\leq k}|$; and

- $h_k|_{|B|} : |B| \rightarrow |B|$ is a homeomorphism for all boxes B in $\mathcal{R}^k, \dots, \mathcal{R}^0$.

We shall construct the homeomorphisms h_k by induction on k . For the base case, $\mathcal{R}^0 \cap A$ is a finite set of points. We define h_0 to be the identity mapping on $|\mathcal{R}^0|$.

Suppose we have constructed a homeomorphism $h_{k-1} : |\mathcal{R}^{\leq k-1}| \rightarrow |\mathcal{R}^{\leq k-1}|$ such that

- $h_{k-1}(A \cap |\mathcal{R}^{\leq k-1}|) = \widehat{A} \cap |\mathcal{R}^{\leq k-1}|$; and
- $h_{k-1}|_{|B|} : |B| \rightarrow |B|$ is a homeomorphism for all boxes in $\mathcal{R}^{k-1}, \dots, \mathcal{R}^0$.

Let $B \in \mathcal{R}^k$, then we define $h_k|_{|B|} : |B| \rightarrow |B|$ as follows. Since the diameter δ of B is smaller than the uniform cone radius of $|B| \cap A$, by Theorem 4.4.1 there exists a homeomorphism $g|_{|B|} : |B| \rightarrow |B|$ such that $g|_{\partial|B|} = id$, and $g|_{|B|}(|B| \cap A) = \text{Cone}(A \cap \partial|B|, \vec{p}_B)$, in case $\vec{p}_B \in \mathcal{P}$, and such that $g|_{|B|}(|B| \cap A) = \text{Cone}(A \cap \partial|B|, \vec{p}_B) - \{\vec{p}_B\}$, in case $\vec{p}_B \in \mathcal{P}'$. We omit this last case since it is completely analogous.

Suppose that $|B| = [a_1, b_1] \times \dots \times [a_n, b_n]$, then $\vec{p}_B = (c_1, \dots, c_n)$, with $c_i = a_i + \frac{b_i - a_i}{2}$ for $i = 1 \dots, n$. We define the following sets

$$|B_t| := [a_1 + (1-t)\frac{b_1 - a_1}{2}, a_1 + (1+t)\frac{b_1 - a_1}{2}] \times \dots \\ \times [a_n + (1-t)\frac{b_n - a_n}{2}, a_n + (1+t)\frac{b_n - a_n}{2}] \quad 0 \leq t \leq 1.$$

Furthermore, we define $f|_{|B|} : |B| \rightarrow |B|$ as

$$f|_{|B|}(\vec{x}) = \partial|B_{t_0}| \cap L',$$

where

- t_0 is such that $\vec{x} \in \partial|B_{t_0}|$,
- L is the half line from \vec{p}_B to \vec{x} ,
- \vec{y} is $L \cap \partial|B|$, and
- L' is the half line from \vec{p}_B to $h_{k-1}(\vec{y})$.

It can easily be verified that $f|_{|B|}$ is a homeomorphism from $|B|$ to $|B|$ such that

$$f|_{|B|}(\text{Cone}(A \cap \partial|B|, (\vec{p}_B)) = \text{Cone}(h_{k-1}((A \cap \partial|B|), \vec{p}_B)). \quad (5.2.1)$$

Finally, we define $h_k|_{|B|} : |B| \rightarrow |B|$ as the composition of the two homeomorphisms $f|_{|B|}$ and $g|_{|B|}$, i.e., $h_k|_{|B|} = f|_{|B|}g|_{|B|}$. Hence $h_k|_{|B|}$ is also a homeomorphism.

We now show that for any k -dimensional box $B' \in \mathcal{R}^k$ in \mathbf{R}^n ,

$$(h_k|_{|B|})|_{|B'|} = (h_k|_{|B'|})|_{|B|}. \quad (5.2.2)$$

Indeed, if $|B| \cap |B'| = \emptyset$, then we are done. Suppose that $\vec{x} \in |B| \cap |B'|$. Then by the definition of a box collection, $\vec{x} \in \partial|B| \cap \partial|B'|$. Now, for every box $B'' \in \mathcal{R}^k$, $h_k|_{\partial|B''|}(\vec{x}) = f|_{\partial|B''|}(\vec{x}) = h_{k-1}(\vec{x})$. Hence,

$$\begin{aligned} (h_k|_{|B|})|_{|B'|}(\vec{x}) &= h_k|_{\partial|B| \cap \partial|B'|}(\vec{x}) \\ &= h_{k-1}(\vec{x}) \\ &= h_k|_{\partial|B'| \cap \partial|B|}(\vec{x}) \\ &= (h_k|_{|B'|})|_{|B|}(\vec{x}). \end{aligned}$$

Since for every two boxes B and B' in \mathcal{R}^k , (5.2.2) is satisfied, we may conclude that $h_k|_{|B| \cup |B'|} = h_k|_{|B|} \cup h_k|_{|B'|} : |B| \cup |B'| \rightarrow |B| \cup |B'|$ is also a homeomorphism (this is known as the Gluing Lemma [52, Lemma 3.8]). We define

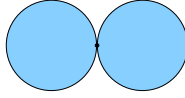
$$h_k := \bigcup_{|B| \in \mathcal{R}^k} h_k|_{|B|}.$$

We still need to verify that $h_k(A \cap |\mathcal{R}^{\leq k}|) = \widehat{A} \cap |\mathcal{R}^{\leq k}|$. It is sufficient to show that $h_k(A \cap |B|) = \widehat{A} \cap |B|$ for any $B \in \mathcal{R}^k$. By (5.2.1), the induction hypothesis, and the definition of \widehat{A} in the algorithm LINEARIZE-IN- n -DIMENSIONS,

$$\begin{aligned} h_k(A \cap |B|) &= \text{Cone}(h_{k-1}(A \cap \partial|B|), \vec{p}_B) \\ &= \text{Cone}(\widehat{A} \cap \partial|B|, \vec{p}_B) \\ &= \widehat{A} \cap |B|. \end{aligned}$$

Since $|\mathcal{R}|$ is closed, using standard techniques from topology [55], the final homeomorphism h_n can be extended to a homeomorphism $h : \mathbf{R}^n \rightarrow \mathbf{R}^n$. \square

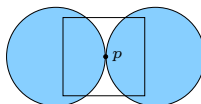
We illustrate the algorithms PREPROCESS and LINEARIZE-IN- n -DIMENSIONS on the following semi-algebraic set in \mathbf{R}^2 :



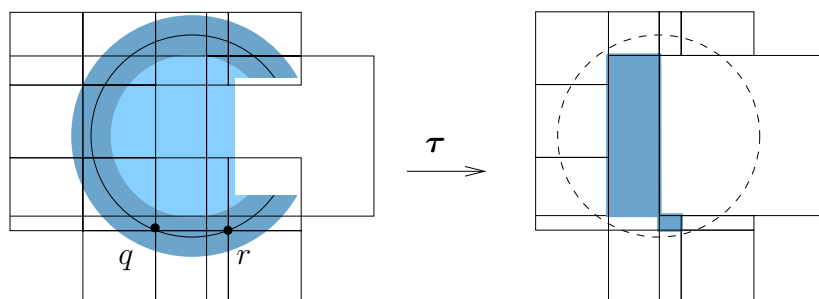
First, we run the algorithm PREPROCESS on it. The output are three regular sets:

$$C_0 = \text{two overlapping blue circles}, \quad C_1 = \text{two overlapping white circles with black outlines}, \quad \text{and } C_2 = \bullet$$

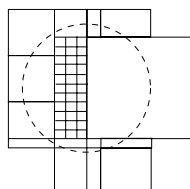
We will only describe the linearization of the left part of the input figure, the right part being a mirrored copy of this. We compute a two-dimensional box collection \mathcal{R} as described in Section 5.1. This is done as follows: compute a box covering of size smaller than the cone radius of C_2 :



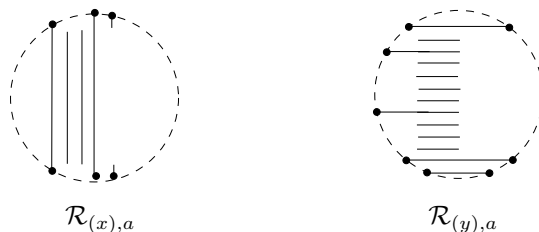
Next, the cone radius δ of the part of C_1 which lies in the dark region of the next figure, is computed. Then, the box covering of size δ of this dark region is computed. As can be seen, C_1 intersects some boxes only at the boundaries (this happens in \vec{q} and \vec{r}). Because we have covered more than just C_1 (more specifically, we covered the dark region), we may translate the box covering with a small translation τ . The resulting box collection is shown in the figure on the right. This box collections in general position with C_0 and C_1 .



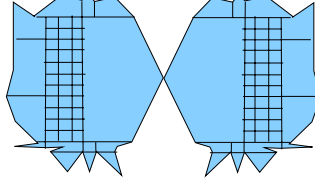
Finally, the dark rectangular two-dimensional part needs to be covered by boxes:



We now have the box collection \mathcal{R} . The next step in the algorithm is the computation of the sets $\mathcal{R}_{(x),a}$ and $\mathcal{R}_{(y),b}$ with $a \in \text{Coord}(\mathcal{R}_{(x)})$ and $b \in \text{Coord}(\mathcal{R}_{(y)})$. These are shown respectively in the left and the right figure:



Since LINEARIZE-IN-1-DIMENSION is trivial, we start directly by connecting the points and line segments with the centers of some of the boxes in \mathcal{R} . By Theorem 5.2.1, this results in a linearization, which is shown in the following figure.



If the linearization obtained in Theorem 5.2.1 also needs to be a good approximation from a metrical point of view, we can easily adapt the algorithms such that the approximation lies arbitrarily close to the original polynomial constraint database. Indeed, we can simply bound the diameter of the boxes used in the construction by a specified ε -value. We will see some applications of these ε -approximations in the next section.

Theorem 5.2.2. *For each n there exists an FO+POLY+TC query ε -approx over the schema $\mathcal{S} = \{S\}$ with S an n -ary relation name, such that for any polynomial constraint database D over \mathcal{S} such that S^D is bounded, ε -approx(D) is an algebraic ε -approximation of S^D . \square*

The general case Let A be an unbounded semi-algebraic set in \mathbf{R}^n . We reduce the construction of an algebraic linearization of A to the construction for bounded semi-algebraic sets as follows:

First, we need to define the *cone radius of A in the point at infinity \vec{p}_∞* . Consider the embedding $i : \mathbf{R}^n \rightarrow \mathbf{R}^{n+1} : (x_1, \dots, x_n) \mapsto (x_1, \dots, x_n, 0)$. Let $\rho : \mathbf{R}^{n+1} \rightarrow \mathbf{R}^{n+1}$ be the reflection defined by $(x_1, \dots, x_{n+1}) \mapsto (x_1, \dots, x_n, -x_{n+1})$. Let $\mathbf{R}^n \cup \{\vec{p}_\infty\}$ be the Alexandrov one-point compactification of \mathbf{R}^n . Finally, consider the stereographic projection $\sigma : S^n((0, \dots, 0), 1) \rightarrow i(\mathbf{R}^n) \cup \{\vec{p}_\infty\}$ defined by $\sigma(x_1, \dots, x_{n+1}) = \frac{(x_1, \dots, x_n)}{1-x_{n+1}}$ and $\sigma(0, \dots, 0, 1) = \vec{p}_\infty$.

We define a cone radius of A at \vec{p}_∞ as a cone radius of the semi-algebraic set

$$i^{-1}(\sigma(\rho(\sigma^{-1}(i(A) \cup \{\vec{p}_\infty\}))))$$

in the origin of \mathbf{R}^n . The local conic structure of semi-algebraic sets implies that there exists an $m > 0$ such that $\{\vec{x} \in \mathbf{R}^n \mid \|\vec{x}\| \geq m\} \cap A$ is topologically equivalent to $\{\lambda \vec{x} \in \mathbf{R}^n \mid \vec{x} \in \partial([-m, m] \times \dots \times [-m, m]) \cap A \wedge \lambda \geq 1\}$.

We now present the unbounded version of the algorithm.

Algorithm LINEARIZE-IN- n -DIMENSIONS':

Input: A semi-algebraic set A in \mathbf{R}^n , and \mathcal{B} an n -dimensional box collection in \mathbf{R}^n .

Output: An \mathbf{A} -linear set \hat{A} in \mathbf{R}^n which is homeomorphic to A .

Method:

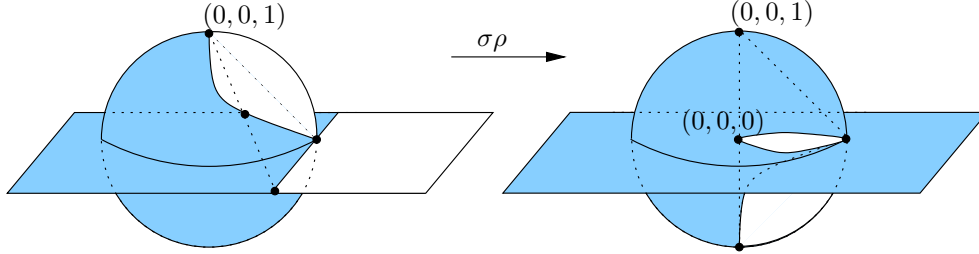


Figure 5.3: A semi-algebraic set is mapped onto the sphere $S^2((0,0,0),1)$, flipped vertically, and projected back onto the sphere $S^2((0,0,0),1)$. This brings the point at infinity \vec{p}_∞ to the origin $(0,0,0)$.

1. Compute a cone radius m of A in \vec{p}_∞ . Let $M = [-m, m] \times \dots \times [-m, m]$.
2. Apply PREPROCESS to $A \cap M$.
3. Apply LINEARIZE-IN- n -DIMENSIONS to the result of step 3 and an empty box collection \mathcal{B} .
4. Output

$$\hat{A} := (\widehat{A \cap M}) \cup \{\lambda \vec{x} \in \mathbf{R}^n \mid \vec{x} \in A \cap \partial M \wedge \lambda \geq 1\}.$$

The following theorem can readily be verified.

Theorem 5.2.3. *For each n there exists an FO+POLY+TC formula `linearize'` over the schema $\mathcal{S} = \{S\}$, with S an n -ary relation name, such that for any polynomial constraint database D over \mathcal{S} , `linearize'`(D) is an algebraic linearization of S^D . \square*

5.3 Rational Linearizations

We now refine the previous theorems to *rational linearization*.

Theorem 5.3.1. *For each n there exists an FO+POLY+TC query `ratlin` over the schema $\mathcal{S} = \{S\}$, with S n -ary, such that for any polynomial constraint database D over \mathcal{S} such that S^D is bounded, `ratlin`(D) is a rational linearization of S^D .*

Proof. We can obtain this result easily by modifying the construction of the special box collection in Section 5.1 in the following way. When in this construction the box covering \mathcal{V} of size $\frac{\epsilon_V}{\sqrt{n}}$ is computed, we compute a rational number that is smaller than $\frac{\epsilon_V}{\sqrt{n}}$, and take this as the size of the box covering \mathcal{V} to be computed. By similar techniques as in Chapter 3, it is easy to show that there exists an FO+POLY+TC query which returns a rational number smaller than the input number. In this way, all boxes in $\mathcal{R} \subset \mathbf{Q}^{2n}$. Hence, the points \vec{p}_B which are selected by the algorithm LINEARIZE-IN- n -DIMENSIONS will have rational coordinates. \square

We also have a rational equivalent of Theorem 5.2.2. Its proof is analogue to the previous one.

Theorem 5.3.2. *For each n there exists an FO+POLY+TC query ε -ratlin over the schema $\mathcal{S} = \{S\}$, with S an n -ary relation name, such that for any polynomial constraint database D over \mathcal{S} such that S^D is bounded, ε -ratlin(D) is a rational ε -approximation of S^D . \square*

5.4 The Connectivity Query

In this section, we show that the connectivity query, which asks whether a polynomial constraint database is connected, is expressible in FO+POLY+TC.

Let A be a semi-algebraic set in \mathbf{R}^n . For semi-algebraic sets, expressing the connectivity query is the same as expressing whether any two points can be connected by a path lying entirely in A [8, Proposition 2.5.13]. One can even choose the paths to be semi-algebraic, in case of a semi-algebraic set, and semi-linear, in case of a semi-linear set [71, Proposition 3.2, Chapter 6].

We now show that this query can be expressed in FO+POLY+TC using the formula $\text{linearize}'$ given in Theorem 5.2.3.

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. Consider the FO+POLY+TC formula $\text{lineconn}(\vec{r}, \vec{s})$ over \mathcal{S}

$$\exists \lambda (0 \leq \lambda \leq 1 \wedge \forall \vec{t} (\vec{t} = \lambda \vec{r} + (1 - \lambda) \vec{s} \rightarrow \text{linearize}'(\vec{t}))).$$

Define the following FO+POLY+TC sentence

$$\text{connected} \equiv \forall \vec{p} \forall \vec{q} \neg \text{linearize}'(\vec{p}) \wedge \text{linearize}'(\vec{q}) \rightarrow [\text{TC}_{\vec{x}, \vec{y}} \text{lineconn}](\vec{p}, \vec{q}).$$

Proposition 5.4.1. *Let $\mathcal{S} = \{S\}$ with S an n -ary relation name. The FO+POLY+TC formula connected asks whether S^D is connected for any polynomial constraint database over \mathcal{S} .*

Proof. Since $\text{linearize}'(D)$ is semi-linear and topologically equivalent to S^D , two points \vec{p} and \vec{q} belong to the same connected component of $\text{linearize}'(D)$ if and only if there exists a semi-linear continuous mapping $\gamma : [0, 1] \rightarrow \mathbf{R}^n$ such that $\gamma(0) = \vec{p}$ and $\gamma(1) = \vec{q}$. This proves the correctness of the formula connected .

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 $\text{linearize}'(D)$, needed to connect any two points in the same connected component of $\text{linearize}'(D)$. Now, any semi-linear set can be decomposed in a finite number of open convex sets. The finiteness of this decomposition yields the desired bound on the number of line segments needed to connect any two points in the same connected component of $\text{linearize}'(D)$. \square

This is the generalization of the well-known result that first-order logic extended with a transitive closure logic can express graph connectivity.

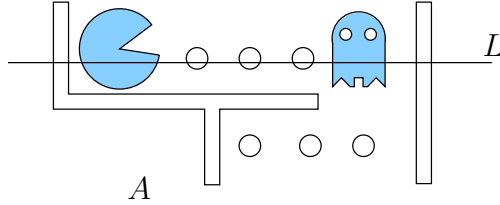


Figure 5.4: A semi-algebraic set A with $\kappa(A) = 12$.

Since FO+POLY+TC is included in stratified DATALOG with polynomial constraints, Proposition 5.4.1 solves the question raised in [23, 48, 50] whether stratified DATALOG with polynomial constraints can express the connectivity of polynomial constraint databases.

5.5 Volume Approximation

In this section, we shall use the box covering and the ε -approximation to approximate the volume of semi-algebraic sets with an FO+POLY+TC formula. We restrict our attention to bounded semi-algebraic sets and require that the evaluation of this FO+POLY+TC formula is effective for all bounded semi-algebraic inputs.

Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. Let D be a polynomial constraint database over \mathcal{S} .

The *volume* of a database D is defined as the Lebesgue-measure of the semi-algebraic set $S^D \subseteq \mathbf{R}^n$, and is denoted by $\text{VOL}(D)$.

Since we want an FO+POLY+TC formula whose evaluation is effective on all databases, it is impossible to define the *exact* volume of polynomial constraint databases in FO+POLY+TC. Indeed, consider the database consisting of the unit disk D in \mathbf{R}^2 . The volume of D equals π . Since π is not algebraic, this value cannot be the output of an effective FO+POLY+TC query.

Hence, as suggested by Koiran [40], and Benedikt and Libkin [7], we consider for each $\varepsilon > 0$, an ε -volume approximation query VOL^ε , such that for any polynomial constraint database D over \mathcal{S} , there exists a real number $v \in \text{VOL}^\varepsilon(D)$ such that

$$|v - \text{VOL}(S^D)| < \varepsilon.$$

It is known that VOL^ε is not expressible in FO+POLY [7]. We show that there exists an FO+POLY+TC expressible ε -volume approximation query.

We will use the following result:

Theorem 5.5.1 ([40]). *Let A be a semi-algebraic set in \mathbf{R}^n , and let δ -cover(A) be its box covering of size δ . Then*

$$|\text{VOL}(A) - \text{VOL}(\delta\text{-cover}(A))| < \frac{1}{\delta}(\text{diam}(A))^{n+1}\kappa(A)n, \quad (5.5.1)$$

where $\kappa(A)$ is the maximal number of connected components of the intersection of A with any axis-parallel line L (see Figure 5.4), and where $\text{diam}(A)$ is the diameter of A .² \square

Theorem 5.5.2. *For each $\varepsilon > 0$, there exists an ε -volume approximation query in FO+POLY+TC.*

Proof. We first show that the number $\kappa(S^D)$ of Theorem 5.5.1 is expressible in FO+POLY+TC. Indeed, first we define n sets K_i which contains $2n - 1$ -tuples $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n, \vec{p})$ where $a_j \in \mathbf{R}$ for $j = 1, \dots, i - 1, i + 1, \dots, n$, and where \vec{p} is either an isolated point on the intersection of A with $\{\vec{x} \mid \bigwedge_{j \neq i} x_j = a_j\}$, or the middle of an interval in this intersection. Using similar techniques as in Chapter 3, we compute for each $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ the number of points \vec{p} , such that $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n, \vec{p}) \in K_i$. We then obtain n sets K'_i consisting of n -tuples $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n, N)$ with $N \in \mathbf{N}$, and we define M_i to be the maximum of all those N which are in K'_i for some $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$. Finally, let $\kappa(S^D) = \max\{M_1, \dots, M_n\}$.

Let $\delta = \frac{1}{\varepsilon}(\text{diam}(S^D))^n \kappa(S^D)n + 1$. By Proposition 4.7.2, the box covering of S^D of size δ is expressible in FO+POLY+TC. By Theorem 5.5.1, $\text{VOL}(\delta\text{-cover}(S^D))$ approximates the volume of S^D within an ε -error margin.

Recall that $\delta\text{-cover}(S^D)$ is represented as a $2n$ -ary relation. Each $2n$ -tuple corresponds to an n -dimensional box of size δ (see Section 4.6). Let $\mathbf{nrofboxes}(y)$ be the formula

$$[\text{TC}_{\vec{b}, x; \vec{b}', x'} \text{lexicographic}(\vec{b}, \vec{b}') \wedge x' = x + 1](\vec{b}_{\min}, 1, \vec{b}_{\max}, y),$$

where $\text{lexicographic}(\vec{b}, \vec{b}')$ is an FO+POLY formula expressing that \vec{b} is less than \vec{b}' with respect to the lexicographical ordering on tuples in \mathbf{R}^n , and where $\vec{b}_{\min}, \vec{b}_{\max} \in \delta\text{-cover}(S^D)$ is the minimum (respectively maximum) n -tuple in $\delta\text{-cover}(S^D)$ with respect to the lexicographical ordering. Finally, let $N \in \mathbf{R}$ such that $\mathbf{nrofboxes}(N)$ holds. Then we define $\text{VOL}^\varepsilon(v)$ to be the FO+POLY+TC formula which expresses that $v = N\delta^n$. \square

Since the δ -approximation of A is included in the box covering $\delta\text{-cover}(A)$, a better volume approximation can be obtained by using the volume of the δ -approximation instead of the volume of $\delta\text{-cover}(A)$. By the next Theorem, this also gives an FO+POLY+TC expressible ε -approximation query.

It is known that taking the volume of a semi-linear set does not take us out the semi-algebraic setting and that the volume of a semi-linear set can be expressed in the aggregate language FO+POLY+SUM [7]. Benedikt and Libkin [7] have shown that the exact volume of a semi-linear set, even with parameters, can be expressed in this aggregate language. Their proof consists of inductively summing over parameterized finite sets. Since we do not allow the computation of the transitive closure of parameterized relations, we restrict ourselves to a single semi-linear set $A \subset \mathbf{R}^n$.

²Let X be a set in \mathbf{R}^n . The *diameter* of X is defined as $\sup\{\|\vec{x} - \vec{y}\| \mid \vec{x}, \vec{y} \in X\}$.

Theorem 5.5.3. *Let $\mathcal{S} = \{S\}$, with S an n -ary relation name. There exists an FO+POLY+TC formula **volume** over \mathcal{S} , such that $\text{volume}(S^D)$ is the volume of S^D for any linear constraint database D over \mathcal{S} .*

Proof. If $\dim(S^D) < n$, then we define $\text{volume}(x) \equiv x = 0$. Suppose that $\dim(S^D) = n$. Since $\text{VOL}(S^D) = \text{VOL}(\text{cl}(\text{int}(S^D)))$, we actually may assume that S^D is closed and consists entirely of n -dimensional pieces.

It is well-known that S^D is a finite union of closed convex sets c_1, \dots, c_r of a partition of \mathbf{R}^n induced by a finite number of $(n-1)$ -dimensional hyperplanes H_1, \dots, H_s . Vandeuren et al.[75] show that there exists an FO+POLY formula **hyperplanes**(v_1, \dots, v_n, d) such that **hyperplanes**(D) consists of s tuples $(\vec{v}_1, d_1), \dots, (\vec{v}_s, d_s)$ such that $H_i = \{\vec{x} \in \mathbf{R}^n \mid \vec{v}_i \vec{x} = d_i\}$. Moreover, there exists an FO+POLY formula **specialpoints** such that **specialpoints**(D) contains the extremal points of the convex sets c_1, \dots, c_s . Recall that the *extremal points* of a convex set are those points which cannot be written as a linear combination of two other points of the convex set [81].

We now define an FO+POLY+TC formula **unique** over \mathcal{S} such that **unique**(D) consists of points $\vec{p}_1, \dots, \vec{p}_s$ such that $\vec{p}_i \in \text{int}(c_i)$ for $i = 1, \dots, s$. The formula **unique** makes use of the following formulae over \mathcal{S} .

- A formula over \mathcal{S} which computes the barycenter of any n -dimensional simplex obtained as the convex hull of an $(n+1)$ -tuple of points in **specialpoints**(D), i.e.,

$$\begin{aligned} \text{barycenter}(\vec{x}) &\equiv \exists \vec{y}_1 \dots \exists \vec{y}_{n+1} \left(\bigwedge_{i=1}^n \text{specialpoints}(\vec{y}_i) \right. \\ &\quad \left. \wedge x_i = \frac{1}{n+1} ((\vec{y}_1)_i + \dots + (\vec{y}_{n+1})_i) \right). \end{aligned}$$

- A formula **interiors** over \mathcal{S} which computes the interiors of the sets c_1, \dots, c_s , i.e.,

$$\text{interiors}(\vec{x}) \equiv S(\vec{x}) \wedge \neg(\exists \vec{v} \exists d (\text{hyperplanes}(\vec{v}, d) \wedge \vec{v}^T \vec{x} = d)).$$

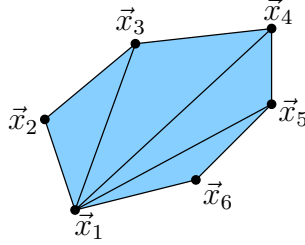
- A formula over \mathcal{S} which checks whether two barycenters are in the same convex set c_i for some i , i.e.,

$$\begin{aligned} \text{samecell}(\vec{x}, \vec{y}) &\equiv \text{barycenter}(\vec{x}) \wedge \text{barycenter}(\vec{y}) \\ &\quad \wedge \forall \lambda (\text{interiors}(\lambda \vec{x} + (1-\lambda)\vec{y}) \wedge 0 \leq \lambda \leq 1). \end{aligned}$$

We then define the formula **unique**(\vec{x}) as

$$\forall \vec{z} [\text{TC}_{\vec{x}; \vec{y}} \text{samecell}] (\vec{x}, \vec{z}) \rightarrow \text{lexicographic}(\vec{x}, \vec{z}),$$

where $\text{lexicographic}(\vec{x}, \vec{z})$ is an FO+POLY formula expressing that \vec{x} is less than \vec{z} with respect to the lexicographical ordering on tuples in \mathbf{R}^n .

Figure 5.5: Triangulation of a convex set in \mathbf{R}^2 .

Define the formula over \mathcal{S}

$$\begin{aligned} \text{extremal}(\vec{x}, \vec{y}) \equiv & \text{specialpoints}(\vec{x}) \wedge \text{unique}(\vec{y}) \\ & \wedge \forall \lambda (\text{interiors}(\vec{y} + \lambda \vec{x}) \wedge 0 \leq \lambda < 1). \end{aligned}$$

We can use this formula together with the points $\vec{p}_1, \dots, \vec{p}_s$ in $\text{unique}(D)$ to identify the extremal points of c_1, \dots, c_s .

It is not difficult to see that there exists an FO+POLY formula $\text{triang}(\vec{x}_1, \dots, \vec{x}_{n+1})$ over \mathcal{S} , such that $\text{triang}(D)$ consist of $(n+1)$ -tuples of points which define a triangulation of S^D for any polynomial constraint database D such that S^D is the set of extremal points of a convex set. By a *triangulation* of a convex set c , we mean a finite set T of $(n+1)$ -tuples of independent points such that

1. $\dim(\text{CH}(\vec{p}_1, \dots, \vec{p}_{n+1}) \cap \text{CH}(\vec{p}'_1, \dots, \vec{p}'_{n+1})) < n$ for all different pairs of $(n+1)$ -tuples of points in T ; and
2. $c = \bigcup_{(\vec{p}_1, \dots, \vec{p}_{n+1}) \in T} \text{CH}(\vec{p}_1, \dots, \vec{p}_{n+1})$.

Here, CH denotes the convex hull of a finite set of points. Let convexhull be the FO+POLY formula over \mathcal{S} , such that $\text{convexhull}(D) = \text{CH}(S^D)$ for any polynomial constraint database such that S^D is a finite set of points. For instance, in the two-dimensional space \mathbf{R}^2 the formula $\text{triang}(\vec{x}_1, \vec{x}_2, \vec{x}_3)$ is defined as the formula which tests the following conditions (this description is taken from [7]): (1) \vec{x}_1 is the lexicographical minimal extremal vertex of the convex set represented by S^D ; (2) either \vec{x}_2 is adjacent to \vec{x}_3 , and \vec{x}_2 is lexicographically less than \vec{x}_3 , and \vec{x}_1 is not adjacent to \vec{x}_2 and \vec{x}_3 , or \vec{x}_1 is adjacent to \vec{x}_2 , and \vec{x}_2 is adjacent to \vec{x}_3 , but \vec{x}_1 is not adjacent to \vec{x}_3 (see Figure 5.5). Here, two points are adjacent if there exists a line on the boundary of the convex set connecting the two points.

Let $(\vec{p}_1, \dots, \vec{p}_{n+1})$ be an $(n+1)$ -tuple of independent points. Let $\vec{r}_i = \vec{p}_i - \vec{p}_1$ for $i = 2, \dots, n+1$, and let G be the $n \times n$ matrix whose rows contain the coordinates of the vectors \vec{r}_j for $1 \leq j \leq n$. Then by the Gram determinant formula [56], the volume of $\text{CH}(\vec{p}_1, \dots, \vec{p}_{n+1})$ is equal to

$$\frac{|\det(GG^t)|^{\frac{1}{2}}}{n!},$$

where G^t is the transpose of G . Hence, the volumes of the convex hulls of $(n+1)$ -tuples of independent points are expressible by an FO+POLY formula, which we will denote by **volsimplex**.

We now define the FO+POLY+TC formula **completetriang** over \mathcal{S} such that **completetriang**(D) is a triangulation of S^D , for any polynomial constraint database D over $\{S\}$.

$$\begin{aligned} \text{completetriang}(\vec{x}_1, \dots, \vec{x}_{n+1}) \equiv \exists \vec{y} (\text{triang}(\text{extremal})(\vec{x}_1, \dots, \vec{x}_{n+1}, \vec{y}) \\ \wedge \text{unique}(\vec{y})). \end{aligned}$$

Next, define

$$\begin{aligned} \Psi(y) \equiv [\text{TC}_{x,s;x',s'} s = \text{volsimplex}(\text{convexhull}(\vec{p}_1, \dots, \vec{p}_{n+1})) \\ \wedge s' = \text{volsimplex}(\text{convexhull}(\vec{q}_1, \dots, \vec{q}_{n+1})) \\ \wedge \text{lexicographic}(\vec{q}_1, \dots, \vec{q}_{n+1}, \vec{p}_1, \dots, \vec{p}_{n+1}) \\ \wedge \text{completetriang}(\vec{p}_1, \dots, \vec{p}_{n+1}) \wedge \text{completetriang}(\vec{q}_1, \dots, \vec{q}_{n+1}) \\ \wedge x' = x + s](0, v_1, y, v_n), \end{aligned}$$

where v_1 and v_n are respectively the volume of the first and last $(n+1)$ -tuple of points according to the lexicographic order. The sum of the volumes of the pieces in the triangulation of S^D is then given by

$$\text{volume}(v) \equiv \exists y \Psi(y) \wedge v = y + v_n,$$

with v_n as above. □

6

The Topological Invariant

In this chapter, we look at semi-algebraic sets from a different point of view. It is known that every semi-algebraic set admits a finite cell decomposition. This cell decomposition can be represented by a finite relational database, which stores cell information and cell adjacencies. This representation is especially important for planar semi-algebraic sets, i.e., sets in \mathbf{R}^2 . In two dimensions, it can be shown that there exists a minimal cell decomposition which characterizes exactly the topology of the semi-algebraic set. This minimal cell decomposition is called the topological invariant.

We are interested in the problem of dynamically maintaining the topological invariant, i.e., we want to keep the topological invariant up-to-date at all times, while the original decomposition is subject to changes. We consider two instances of this problem.

First, we represent semi-algebraic sets as labeled plane graphs, and describe a fully dynamic algorithm built on top of the doubly-connected edge list data structure. This algorithm reacts correctly to all kinds of updates on labeled plane graphs and has linear time complexity per update.

Secondly, we omit the planarity restriction and consider arbitrary graphs. We define the topological invariant of a graph and describe the problem of maintaining the topological invariant of a graph online. Two partially dynamic algorithms are presented. They react correctly to edge insertion only, and have both logarithmic time complexity per update. We conclude this chapter with an experimental evaluation of the relative performance of these two algorithms.

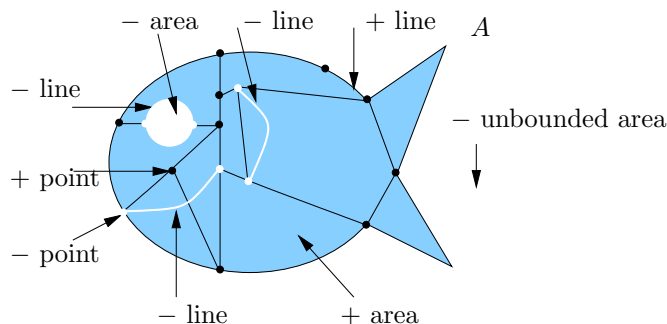


Figure 6.1: A planar semi-algebraic set A and a possible partition of the plane corresponding to A .

6.1 The Maintenance of the Topological Invariant of Labeled Plane Graphs

6.1.1 Definitions

A *plane graph*, geometrically speaking, is a planar embedding of a planar graph. Viewed purely combinatorially, however, we define a plane graph as a structure consisting of *points*, *lines*, and *areas*, and the following associations:

- to each point, a circular list of its outgoing lines in clockwise order;
- to each line, its starting point, its “twin” line, and the area to its left; and
- to each area, the set of isolated points lying in that area.

The *twin line* represents the same geometric line, but viewed from the other direction. So every geometric line is represented by a pair of two combinatorial line objects, one for each direction. The unbounded area is specifically designated as such.

Suppose we augment a plane graph with a $+$ or $-$ label for each point, line, and area. We call this structure a *labeled plane graph*. If we also add algebraic coordinates for each point, and a semi-algebraic definition of a curve for each line, we call the resulting structure a *concrete labeled plane graph*.

A semi-algebraic set A can be represented by a concrete labeled plane graph as follows (see Figure 6.1). Consider a partition of the plane such that

- each class is of one of the following four types: a single point; homeomorphic to \mathbf{R} ; to \mathbf{R}^2 ; or to \mathbf{R}^2 minus a finite number of points;
- exactly one class is unbounded, and must have dimension 2;
- each class is labeled by $+$ or $-$;
- the union of all $+$ -labeled classes equals S .

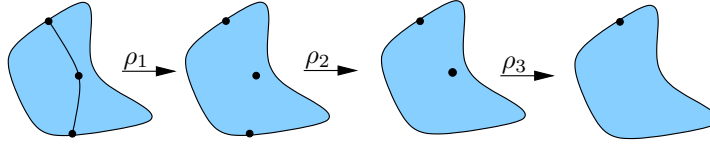


Figure 6.2: The three simplification rules ρ_1 , ρ_2 , and ρ_3

It is known that every semi-algebraic set admits such decomposition [42, 62] and the complexity of computing such decomposition is in NC .

This partition naturally yields a concrete labeled plane graph representing A . Note that in general, there can be many concrete labeled plane graphs representing S . However, one concrete labeled plane graph represents only one semi-algebraic set.

Take a concrete labeled plane graph G and a semi-algebraic set A such that G represents S . Let H be the labeled plane graph underlying G . Then we say that H also represents A . Note that a labeled plane graph can represent many different semi-algebraic sets. However, we will see next that all these sets must be isotopic. The two sets A and A' are called *isotopic* if there exists an orientation-preserving homeomorphism $h : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ such that $h(A) = A'$.

Call two labeled plane graphs H_1 and H_2 *equivalent* if they represent precisely the same semi-algebraic sets. We will now show how we can associate to each labeled plane an equivalent one, which is unique up to isomorphism, and is such that none of the following patterns occur in it:

- a line with the same label as its left and right areas (the right area of a line is the left area of its twin);
- a point with only two outgoing lines, which have the same label as the point;
- an isolated point with the same label as its area .

We then call a labeled plane graph a *topological invariant*. The following theorem shows the importance of topological invariants:

Theorem 6.1.1 ([46]). *Let H and H' be topological invariants and let A and A' be semi-algebraic sets represented by H and H' respectively. Then H and H' are isomorphic if and only if A and A' are isotopic. \square*

The topological invariant can also be defined such that Theorem 6.1.1 holds for homeomorphisms instead of isotopies. This definition of the topological invariant is used by Papadimitriou, Suciu and Vianu [57].

Note that the if-implication in the above theorem implies that for each labeled plane graph there is a topological invariant. Indeed, by the theorem, there cannot be two that are non-isomorphic. Moreover, to find the equivalent topological invariants, we can use the following simplification rules (see Figure 6.2):

- ρ_1 : if there is a line with the same label as its left and right areas, remove this line and merge the two areas.

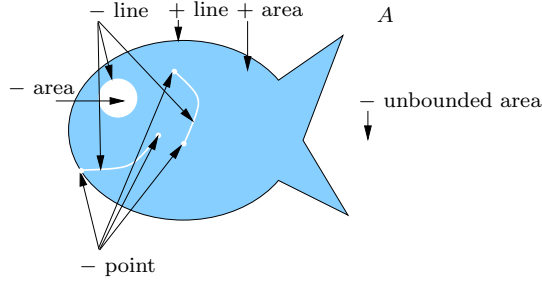


Figure 6.3: The maximal partition of \mathbf{R}^2 induced by the semi-algebraic set A , shown in Figure 6.1. The labeled planar graph describing this decomposition is the topological invariant of A

ρ_2 : if there is a point with only two outgoing lines, which have the same label as the point, remove this point and merge the two lines. (do this only when the two lines are not twins of each other).

ρ_3 : if there is an isolated point with the same label as its area, remove this point.

It can be verified that the rewrite system $\{\rho_1, \rho_2, \rho_3\}$ is terminating and has the Church-Rosser property. Moreover, by first applying ρ_1 exhaustively, then ρ_2 , and finally ρ_3 , we can transform an arbitrary labeled plane graph into its topological invariant in linear time.

The *topological invariant of a planar semi-algebraic set A* is a topological invariant G such that A is represented by G (see Figure 6.3).

6.1.2 Data Structure and Updates

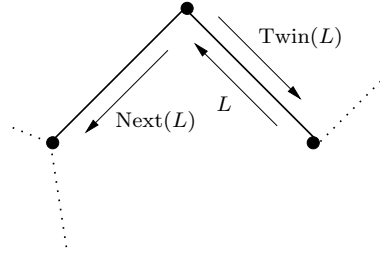
From the definition of a labeled plane graph in Section 6.1, it is straightforward to use the *doubly-connected edge list* [15, 60] as data structure for labeled plane graphs, after two remarks have been made. (1) In order to associate the circular list of outgoing lines to a point, we store for each point a pointer to an arbitrary outgoing line record; (2) To efficiently find the lines on the boundary of an area, we store for each area a pointer to an arbitrary line on the outer boundary of the area, and a list of pointers to arbitrary lines on the inner boundaries of the area.

Furthermore, we extend each record with a label (+ or -), and in case of a concrete labeled plane graph, each point record is extended with coordinate fields, and each line record contains its semi-algebraic definition.

We also need the notion of the “*Next*” of a line L : this is the next line in the circular list of the source of the twin line of L , \bar{L} . This corresponds to walking on the outer boundary of the area in counterclockwise direction (see Figure 6.4).

We now introduce our update dictionary.

1. Insert an isolated point p to area α .

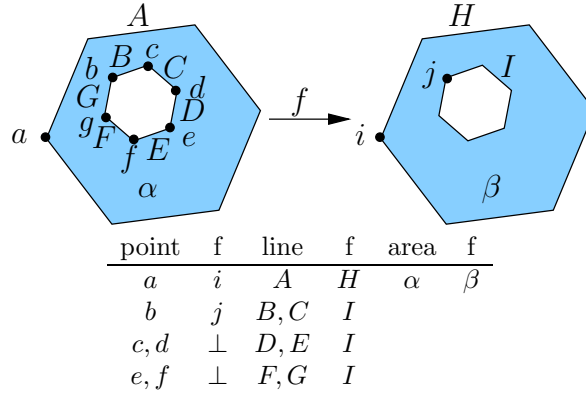
Figure 6.4: The Next and Twin of a line L

2. Insert a point p to line L , splitting the line L into two lines L_1 and L_2 .
3. Insert a line N between two points p and q , following the line L in the circular list of outgoing lines of p , and preceding line M in the circular list of outgoing lines of q . (L , respectively M , does not need to be specified if p , respectively q , is isolated.)
4. Delete isolated point p from area α . This is only allowed if the label of point p equals the label of area α .
5. Delete point p with only two outgoing lines $\overline{L_1}$ and L_2 , coalescing lines L_1 and L_2 into a new line L . This is only allowed if L_1 and L_2 have the same label as p .
6. Delete line L , coalescing its adjacent areas. This is only allowed if the areas adjacent to L have the same label as L .
7. Change the label of point p .
8. Change the label of line L .
9. Change the label of area α .

The implementation of updates on doubly-connected edge lists is trivial for updates 7, 8, and 9, and is straightforward for updates 1, 2, 4, 5, and 6. The only update that needs further comment is update 3. When a line N is added, it splits an area α in two, possibly the same, areas α_1 and α_2 . It depends on the existence of a path between p and q on the boundary of α , whether α_1 equals α_2 , or not. To decide if a path exists, start with line L and apply Next repeatedly. If \overline{M} is reached, a path between p and q exists. If L is reached, no path exists between p and q . In the first case α_1 does not equal α_2 , while in the second case α_1 equals α_2 .

6.1.3 The Maintenance Algorithm

Given two doubly-connected edge lists: one representing the concrete labeled plane graph, and one representing the topological invariant. If we perform one of the nine

Figure 6.5: Example of the partial function f

updates of Section 6.1.2 on the concrete labeled plane graph, what has to be done on the topological invariant such that it remains the topological invariant of the changed concrete labeled plane graph?

To obtain a better performance than computing the topological invariant of the concrete labeled plane graph from scratch, we maintain a correspondence between the concrete labeled plane graph and its topological invariant. This correspondence is given by the partial function f as is illustrated in Figure 6.5.

The partial function f is either defined for an object o of the concrete labeled plane graph, or is undefined in o , in which case we write $f(o) = \perp$. The function f is surjective and is always defined on areas.

The first auxiliary notion we introduce is that of the *coalesce class* of a line L or area α in the concrete labeled plane graph. This is defined as the set of all lines L' (areas α') for which $f(L') = f(L)$ ($f(\alpha') = f(\alpha)$). Of course the crucial issue is how to find this coalesce class. This is possible in time proportional to the size of the coalesce class, and will govern the complexity of our algorithm as is shown in Theorem 6.1.2

Translating the Simplification Rules In our maintenance algorithm, we will need to apply the simplification rules ρ_1 , ρ_2 and ρ_3 defined in Section 6.1.1 to specified parts of the topological invariant. In doing so, we also must update our mapping f . The details are as follows:

ρ_1 : Let L be a line in the concrete labeled plane graph, and let α and β be its adjacent areas. Assume that $f(L)$ is defined, and has the same label as $f(\alpha)$ and $f(\beta)$. We then perform the following operations:

- (a) Delete $f(L)$ from the topological invariant. This will imply that $f(\alpha)$ and $f(\beta)$ will be coalesced in the topological invariant into an area γ .
- (b) Put $f(L) := \perp$.

(c) Put $f(\alpha') := \gamma$ for each α' in the coalesce class of α , and similarly for β .

ρ_2 : Let p be a point in the concrete labeled plane graph, and let L and M be its only adjacent lines. Assume that $f(p)$, $f(L)$, and $f(M)$ are defined, and all have the same label. We then perform the following operations:

(a) Delete $f(p)$ from the topological invariant. This will imply that $f(L)$ and $f(M)$ will be coalesced in the topological invariant into a line N .

(b) Put $f(p) := \perp$.

(c) Put $f(L') := N$ for each L' in the coalesce class of L , and similarly for M .

ρ_3 : Let p be an isolated point in the concrete labeled plane graph, and adjacent with the area α . Assume that $f(p)$ is defined, and $f(p)$ and $f(\alpha)$ have the same label. We then perform the following operation:

(a) Delete $f(p)$ from the topological invariant.

(b) Put $f(p) := \perp$.

Translating the Inverse Simplification Rules In our maintenance algorithm, we will also need to apply “inverses” of the simplification rules. We denote these procedures by δ_1 , δ_2 , and δ_3 . They work as follows:

δ_1 : Let N be a line of the concrete labeled plane graph with endpoints p and q , predecessor L in the list of outgoing lines of p , and successor M in the list of outgoing lines of q . Assume that both $f(p)$ and $f(q)$ are defined, and $f(N) = \perp$. Denote with α_1 and α_2 the areas adjacent to L . In this case $f(\alpha_1) = f(\alpha_2) = \gamma$. We then perform the following operations:

(a) In the circular list around p , starting with L , look backwards for a line L' such that $f(L')$ is defined. Similarly, in the circular list around q , starting with M , look forwards for a line M' such that $f(M')$ is defined.

(b) Insert a new line, $f(N)$, in the topological invariant between $f(p)$ and $f(q)$ with predecessor $f(L')$ and successor $f(M')$.

(c) By the insertion of $f(N)$ in the topological invariant, we have possibly split area γ in two areas γ_1 and γ_2 . We partition the coalesce class of α_1 (which coincides with coalesce class of α_2) as follows: If α'_1 is in the coalesce class of α_1 , and there exists a path connecting a point in α_1 to a point in α'_1 , such that this path only crosses lines $K \neq N$, for which $f(K) = \perp$, then α'_1 is in the new coalesce class of α_1 . Put $f(\alpha'_1) = \gamma_1$ and similarly for α_2 .

δ_2 : Let p be a point in the concrete labeled plane graph, and L and M its only two different outgoing lines. Assume that $f(p) = \perp$, and $f(L) = f(M) = N$. We then perform the following operations:

(a) Insert a new point, $f(p)$, in the topological invariant on line N .

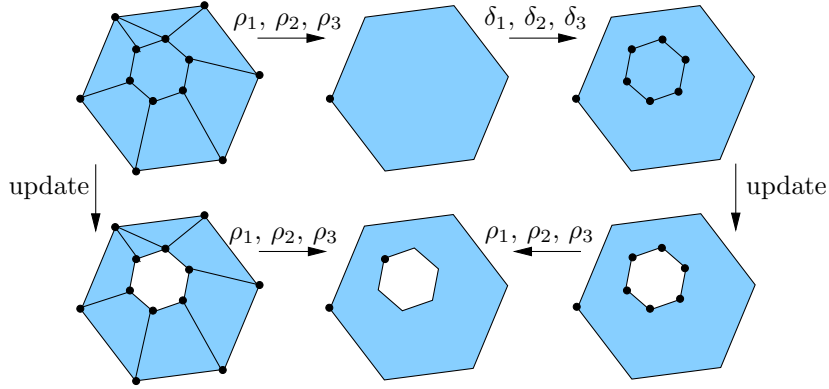


Figure 6.6: The fully dynamical maintenance algorithm

- (b) By the insertion of $f(p)$ in the topological invariant, the line N is split in two lines N_1 and N_2 . We partition the coalesce class of L (which equals the coalesce class of M) as follows: If L' is in the coalesce class of L and is reachable from L by repeatedly performing Next, then L' is in the new coalesce class of L . Put $f(L') = N_1$ and similarly for M .

δ_3 : Let p be a point of the concrete labeled plane graph such that either p is isolated, or f is undefined for all outgoing edges of p . Let α be an adjacent area of p and assume that $f(p) = \perp$. We then perform the following operation:

- (a) Insert a new isolated point, $f(p)$, in the topological invariant to area $f(\alpha)$.

All the procedures ρ_i and δ_i change the topological invariant and adapt the partial function f to this new topological invariant. The difference between ρ_i and δ_i is that the parameters of ρ_i are objects of the topological invariant, while for δ_i , the parameters are objects of the concrete labeled plane graph.

The Maintenance Algorithm We are now ready to describe the maintenance algorithm (see Figure 6.6). It is a *fully dynamic algorithm*, which means that it keeps the topological invariant up-to-date when the concrete labeled plane graph is subject to one of nine updates defined in Section 6.1.2.

Consider the nine kinds of updates defined in Section 6.1.2.

1. *Insertion of a new isolated point p in area α .*
 - (a) Insert a new point, $f(p)$, in area $f(\alpha)$ of the topological invariant.
2. *Insertion of a new point p on a line L .*
 - (a) If $f(L) = \perp$ and $f(r) = \perp$ and/or $f(s) = \perp$, where r and s are the endpoints of L , perform δ_3 or δ_2 on r and/or s , and apply δ_1 on L .

- (b) Insert a new point, $f(p)$, in the topological invariant on $f(L)$.
 - (c) By the insertion of p in the concrete labeled graph, the line L is split in two lines L_1 and L_2 . Correspondingly, the line $f(L)$ is split into M_1 and M_2 . We partition the coalesce class of $\overline{L_1}$ (which equals the coalesce class of L_1 and L_2) as follows: if L'_1 is in the coalesce class of $\overline{L_1}$ and is reached by performing Next on $\overline{L_1}$, then L'_1 is in the new coalesce class of $\overline{L_1}$. Put $f(L'_1) = M_1$ and similarly for L_2 .
 - (d) Finally apply ρ_1 to M_1 and M_2 if possible, and perform ρ_2 or ρ_3 to $f(p)$, $f(r)$, and $f(s)$, if possible
3. *Insertion of a new line N between p and q with predecessor L in the list of lines around p , and successor M in the list around q .*
- (a) If $f(p) = \perp$, perform δ_2 or δ_3 (whichever appropriate) on p . Similarly for q .
 - (b) In the circular list around p , starting with L , look backwards for a line L' such that $f(L')$ is defined. Similarly, in the circular list around q , starting with M , look forwards for a line M' such that $f(M')$ is defined.
 - (c) Insert a new line, $f(N)$, in the topological invariant between $f(p)$ and $f(q)$ with predecessor $f(L')$ and successor $f(M')$.
 - (d) By the insertion of N in the concrete labeled plane graph, we have possibly split an area α in two areas α_1 and α_2 . Correspondingly, $f(\alpha)$ has been split in two areas γ_1 and γ_2 . We partition the coalesce class of α_1 (which coincides with coalesce class of α_2) as follows: If α'_1 is in the coalesce class of α_1 , and there exists a path connecting a point in α_1 to a point in α'_1 , such that this path only crosses lines $K \neq N$, for which $f(K) = \perp$, then α'_1 is in the new coalesce class of α_1 . Put $f(\alpha'_1) = \gamma_1$ and similarly for α_2 .
 - (e) Finally, perform ρ_1 on $f(N)$ if possible, and perform ρ_2 and ρ_3 on $f(p)$ and $f(q)$, if possible.
4. *Deletion of an isolated point p in area α .*
- (a) This update is already performed by ρ_3 on the topological invariant, if possible.
5. *Deletion of point p with only two outgoing lines $\overline{L_1}$ and L_2 .*
- (a) This update is already performed by ρ_2 on the topological invariant, if possible
 - (b) By the deletion of point p , we have coalesced the lines L_1 and L_2 into a new line, L , in the concrete labeled plane graph. Set $f(L) := f(L_1) = f(L_2)$.
6. *Deletion of a line L with endpoints p and q .*
- (a) This update is already performed by ρ_1 on the topological invariant, if possible.

- (b) By the deletion of the line L , we have possibly coalesced the adjacent areas α_1 and α_2 into a new area, α , in the concrete labeled plane graph. Set $f(\alpha) := f(\alpha_1) = f(\alpha_2)$.

7. *Relabeling a point p .*

- (a) If $f(p) = \perp$, and $f(L') = \perp$ for each L' in the circular list around p , then apply δ_3 to p . If $f(p) = \perp$, and $f(L') = \perp$ for each L' in the circular list around p , except for two lines $\overline{L_1}$, and L_2 , such that $f(L_1) = f(L_2)$, apply δ_2 to p .
- (b) Relabel point $f(p)$.
- (c) Apply ρ_2 or ρ_3 to $f(p)$, if possible.

8. *Relabeling a line L between p and q .*

- (a) If $f(p) = \perp$, perform δ_2 or δ_3 (whichever appropriate) on p and similarly for q .
- (b) If $f(L) = \perp$, then apply δ_1 to line L .
- (c) Relabel $f(L)$.
- (d) Perform ρ_1 to $f(L)$ if possible, and apply ρ_1 or ρ_2 to $f(p)$ and $f(q)$, if possible.

9. *Relabeling an area α .*

- (a) If $f(p) = \perp$ for an isolated point p in α , or a point p on the boundary of α , for which f is undefined for all outgoing lines, then perform δ_3 to p . If $f(q) = \perp$ for a point q on the boundary of α , then apply δ_2 to q . Finally, if $f(L) = \perp$ for a line L on the boundary of α , then apply δ_1 to L .
- (b) Relabel area $f(\alpha)$.
- (c) Apply ρ_1 for each line $f(L)$ on the boundary of $f(\alpha)$, if possible. For each point $f(q)$ on the boundary of $f(\alpha)$, perform ρ_2 if possible, and for each isolated point $f(p)$ in $f(\alpha)$, apply ρ_3 , if possible.

Complexity Analysis The time needed to maintain the topological invariant by the Maintenance Algorithm just described is equal to the size of the coalesce classes involved. Since these coalesce classes can be linear in the size of the topological invariant, we obtain that the Maintenance Algorithm has linear time complexity per update.

Theorem 6.1.2. *The total time spent on ℓ updates by Maintenance Algorithm is $O(\ell^2)$.*

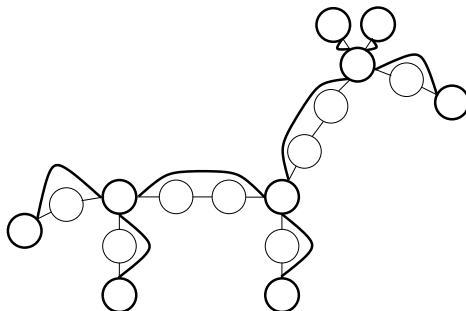


Figure 6.7: A graph (thin lines) together with its topological invariant (thick lines).

6.2 The Maintenance of the Topological Invariant of Arbitrary Graphs

6.2.1 Definitions

We can also define the notion of a topological invariant for arbitrary graphs. Consider an undirected graph with weighted edges $G = (V, E, \lambda)$, with $n = |V|$ the number of vertices, and $m = |E|$ the number of edges. The weights of the edges are given by a mapping $\lambda : E \rightarrow \mathbf{N}^+$. Two vertices are adjacent if there exists an edge between them. We will use the following definitions

1. A vertex v is *regular* if and only if it is adjacent to precisely two edges.
2. A vertex that is not regular is called *singular*.
3. A path between two singular vertices that passes only through regular vertices is called a *regular path*.

We assume that the graph G does not contain regular cycles, i.e., a cycle consisting only of regular vertices.

The *topological invariant* $G_I = (V_I, E_I, \lambda_I)$ of the graph G , is a multigraph which is obtained as follows (see Figure 6.7).

1. V_I is the subset of V consisting of all singular vertices.
2. E_I consists of all pairs (v, w) such that there is a regular path between v and w . There is an edge for each regular path between v and w . We call these edges *topological edges*.
3. $\lambda_I((v, w))$ is the sum of all $\lambda(e)$ where e is an edge on the regular path between v and w , corresponding to the edge (v, w) .

6.2.2 Data Structure and Updates

We will use the standard data structures for undirected graphs, as provided e.g., by LEDA (Library of Efficient Data Types and Algorithms,[54]). Since these data structures are standard, we discuss how the updates on the graph G must be translated into updates on its topological invariant G_I .

We only consider insertions of a new isolated vertex and insertions of edges between existing vertices in the graph G (other more complex insertion operations can be translated into a sequence of these basic insertion operations). The insertion of an isolated vertex is handled trivially, i.e., we insert it in V_I .

For the insertion of an edge we distinguish between six cases that are explained below and depicted in Figures 2–7. The left side of each figure shows the situation before the insertion of the edge $\{x, y\}$, which is the dotted line and the right side shows the situation after the insertion. As before, the topological edges are drawn in thick lines.

Case 1 Vertices x and y are both singular and $\deg(x) \neq 1$ and $\deg(y) \neq 1$. Then the edge $\{x, y\}$ is also *inserted* in G_I .

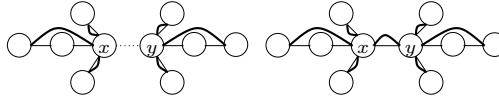


Figure 2: Case 1

Case 2 Vertices x and y are both singular and one of them, say x , has degree one. Let $\{z, x\}$ be the edge in G_I adjacent to x . *Extend* this edge to the new edge $\{z, y\}$ in G_I , putting $\lambda_I(\{z, y\}) := \lambda_I(\{z, x\}) + \lambda(\{x, y\})$. Note that x becomes a regular vertex after the insertion.

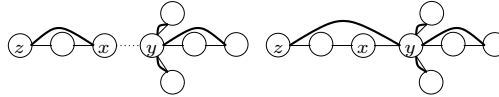


Figure 3: Case 2

Case 3 Vertices x and y are both singular and $\deg(x) = \deg(y) = 1$. Let $\{z_1, x\}$ ($\{z_2, y\}$) be the edge in G_I adjacent with x (y). Suppose $z_1 \neq y$ ($z_2 \neq x$). Then *merge* the edges $\{z_1, x\}$ and $\{y, z_2\}$ in G_I into a single, new edge $\{z_1, z_2\}$ in G_I , putting $\lambda_I(\{z_1, z_2\}) := \lambda_I(\{z_1, x\}) + \lambda_I(\{y, z_2\}) + \lambda(\{x, y\})$. If $z_1 = y$ ($z_2 = x$) then *extend* the edge $\{x, y\}$ to, a new edge $\{x, x\}$ in G_I , putting $\lambda_I(\{x, x\}) := \lambda_I(\{x, y\}) + \lambda(\{x, y\})$.

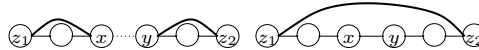


Figure 4: Case 3

Case 4 One of the vertices x and y is regular, say x , and the other vertex, y , is singular and has degree one.

First, the edge $\{z_1, z_2\}$ of G_I which corresponds to the regular path between z_1 and z_2 on which x lies, must be *split* into two new edges $\{z_1, x\}$ and $\{x, z_2\}$ of G_I . Here, we put $\lambda_I(\{z_1, x\}) := \sum \lambda(\{u, v\})$, where the summation is over all edges in G on the regular path from z_1 to x . We similarly define $\lambda_I(\{x, z_2\})$. Secondly, let $\{z_3, y\}$ be the edge in G_I adjacent to y . Then we *extend* this edge to a new edge $\{z_3, x\}$ in G_I , putting, $\lambda_I(\{x, z_3\}) := \lambda_I(\{y, z_3\}) + \lambda(\{x, y\})$.

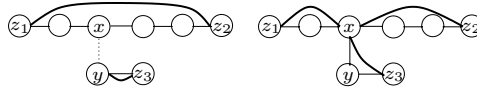


Figure 5: Case 4

Case 5 One of the vertices, say x , is regular and the other one, y , is singular with degree not equal to one.

Then we split exactly as in case 4, and now we also insert $\{x, y\}$ as a new edge in G_I .

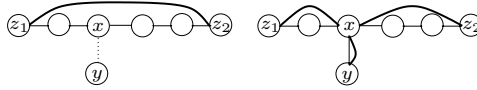


Figure 6: Case 5

Case 6 Both x and y are regular: two *splits* must be performed

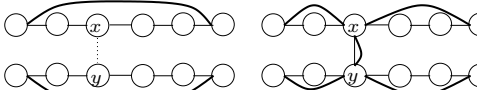


Figure 7: Case 6

As can be seen in the description of cases 1–6, if no regular vertices are involved, then the update on the graph G translates trivially in exactly the same update on the topological invariant G_I . It is only in cases 4, 5, and 6, that the update on the graph G involves vertices which *have no counterpart* in the topological invariant G_I . However, we need to know which edge we need to split and what the weights are of the topological edges created by the split. Consequently, the problem of maintaining the topological invariant G_I of a graph G amounts to two tasks:

- Maintain a function *find topological edge*, which takes a regular vertex as input, and outputs the topological edge whose regular path in G contains the input vertex.
- Maintain a function *find weights* which outputs the weights of the edges created when a topological edge is split at the input vertex.

We note that the Maintenance Algorithm of Section 6.1.3 finds the topological edge, by storing, for each regular vertex, a pointer to its topological edge. This makes

the topological edge accessible in constant time, but the maintenance of the pointers under updates can cost linear time in the worst case.

We next describe two algorithms which are more efficient. Both algorithms keep the topological invariant of a graph up-to-date when the graph is subject to edge insertions only.

6.2.3 The Renumbering Algorithm

We first show how the topological edges can be found efficiently.

Assigning numbers to the regular vertices We number the regular vertices, that lie on a regular path, consecutively. The numbers of the regular vertices on any regular path will always form an interval of the natural numbers. The Renumbering Algorithm will maintain two properties:

Interval property the assignment of *consecutive* numbers to *consecutive* regular points;

Disjointness property *different* regular paths have *disjoint* intervals.

We then have a unique interval associated with each regular path, and hence with each topological edge of size > 0 . Moreover, we choose the minimum of such an interval as a unique number associated with a topological edge. Specifically, the minimal number serves as a *key* in a *dictionary*. Recall that in general, a dictionary consists of pairs $\langle \text{key}, \text{item} \rangle$, where the item is unique for each key. Given a number k , the function which returns the item with the maximal key smaller than k can be implemented in $O(\log N)$ time, where N is the number of items in the dictionary [14].

The items we use contain the following information.

1. An identifier of the topological edge associated with the key.
2. The number of regular vertices on the regular path corresponding to this topological edge.
3. An identifier of the regular vertex on the regular path corresponding to this topological edge, which has the key as number.

In Figure 6.8 we give an example of a dictionary containing three keys, corresponding to the three topological edges in the topological invariant G_I of the graph G .

Maintaining the numbers of the regular vertices We must now show how to maintain this numbering under updates, such that the interval and disjointness properties mentioned above remain satisfied.

Actually, only in case 3 in Section 6.2.2 we need to do some maintenance work on the numbering. Indeed, by merging two topological edges, the numbering of the regular vertices is no longer necessarily consecutive. We resolve this by *renumbering*

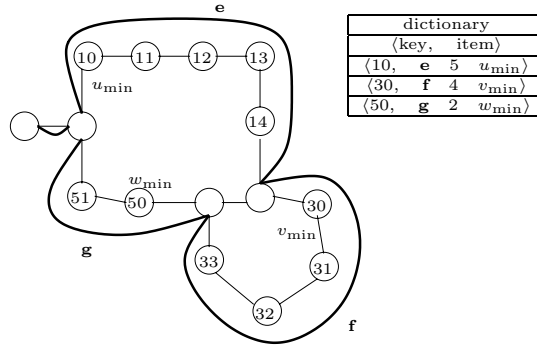


Figure 6.8: Dictionary example.

the vertices on the shortest of the two regular paths. Note that the size of a regular path is stored in the dictionary item for that path.

In order to keep the intervals disjoint, we must assume that the maximal number of edge insertions to which we need to respond is known in advance. Concretely, let us assume that we have to react to at most ℓ update operations.¹ A regular path is “born” with at most two regular vertices on it. Every time a new regular path is created, say the k th time, we assign the number $2k\ell$ to one of the two regular vertices on it. Hence, newly created topological edges correspond to numbers which are 2ℓ apart from each other. Since a newly created topological edge can become at most $\ell - 1$ vertices longer, no interference is possible.

Finding the topological edge Consider that we are in one of the cases 4–6 described in Section 6.2.2, where we have to split the topological edge at vertex x . We look at the number of x , say k , and find in the dictionary the item associated with the maximal key smaller than k . This key corresponds to the interval to which k belongs, or equivalently, to the regular path to which x belongs. In this way we find the topological edge which has to be split, since this edge is identified in the returned item.

The numbering thus enables us to find an edge in $O(\log m')$ time, where m' is the number of edges in G_I which correspond to a regular path passing through at least one regular vertex. Because m' is at most m , the number of edges in G , we obtain:

Proposition 6.2.1. *Given a regular vertex and its number, the dictionary returns in $O(\log m)$ time the topological edge corresponding to the regular path on which this regular vertex lies.*

We next show how, when a topological edge is split, we can quickly find the weights of the two new edges created by the split.

¹This assumption is rather harmless: one can set this maximum limit to a large number. If it is eventually reached, we restart from scratch.

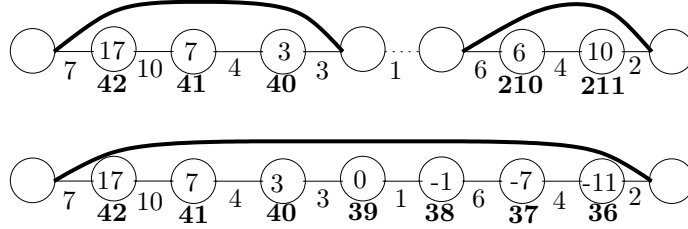


Figure 6.9: Assigning new numbers and weights of regular vertices simultaneously when two topological edges are merged. The numbers of regular vertices are in bold, the weights are inside the vertices.

Assigning weights to the regular vertices We define the *weight* of a regular vertex as a mapping $\lambda^* : V_r \rightarrow \mathbf{N}$ and assign values to this mapping. Let us define the k th vertex of a regular path as the vertex with number s , such that $s - s_{\min} = k$, where s_{\min} is the minimal number of the vertices on the regular path. Let v_k be the k th regular vertex, and let $\{x, y\}$ be the topological edge corresponding to the regular path. We then define $\lambda^*(v_0) := 0$ for $k = 0$, and $\lambda^*(v_k) := \lambda^*(v_{k-1}) + \lambda(\{v_{k-1}, v_k\})$ for $k > 0$.

Maintaining the weights of regular vertices The maintenance of the weights of regular vertices, under edge insertions is easy. It requires only constant time when a topological edge is extended. Let $\{x, y\}$ be a topological edge, and suppose that we extend this edge by inserting $\{y, z\}$. Let u be the regular vertex adjacent to y . Then,

- if $\lambda^*(u) < 0$, then $\lambda^*(y) := \lambda^*(u) - \lambda(\{u, y\})$.
- if $\lambda^*(u) \geq 0$, and no regular vertex with a positive weight is adjacent to u , then $\lambda^*(y) := \lambda^*(u) + \lambda(\{u, y\})$. Otherwise, the $\lambda^*(y) := \lambda^*(u) - \lambda(\{u, y\})$.

It takes no time at all when a topological edge is split. However, when two topological edges are merged, we need to adjust the weights of the regular vertices on the shortest of the two regular paths, as shown in Figure 6.9. This adjustment of the weights can clearly be done simultaneously with the renumbering of the vertices, as explained above.

Finding the weights The weights of regular vertices now enable us to find the weights of the two edges created by a split of a topological edge in logarithmic time. Indeed, given the number of the regular vertex where the split occurs, we search in the dictionary which topological edge needs to be split; call it $\{z_1, z_2\}$. In the returned item we find the vertex which has the minimal number of the vertices on the regular path corresponding to $\{z_1, z_2\}$. Denote this vertex with u which is adjacent to either z_1 or z_2 . We assume that u is adjacent to z_1 , the other case being analogous. The weight of the two new topological edges $\{z_1, x\}$ and $\{x, z_2\}$ can be computed easily:

- $\lambda(\{z_1, x\}) := \lambda(\{z_1, u\}) + |\lambda^*(u) - \lambda^*(x)|$; and

- $\lambda(\{x, z_2\}) := \lambda(\{z_1, z_2\}) - \lambda(\{z_1, x\})$.

If only one regular vertex remains on a regular path after a split, or a regular vertex becomes singular, then the weight of this vertex is set to 0. This all takes constant time plus the time for one lookup in the dictionary, which takes logarithmic time. Hence, with ℓ the current number of edge insertions, we obtain:

Proposition 6.2.2. *The weights of the two new edges created by a split can be computed in $O(\log \ell)$ time.*

Complexity analysis By the *amortized complexity* of an on-line algorithm [68, 53], we mean the total computational complexity of supporting ℓ updates (starting from the empty graph), as a function of ℓ , divided by ℓ to get the average time spent on supporting one single update. We will prove here that the Renumbering Algorithm has $O(\log \ell)$ amortized time complexity. We only count edge insertions because the insertion of an isolated vertex has zero cost.

Theorem 6.2.1. *The total time spent on ℓ updates by the Renumbering Algorithm is $O(\ell \log \ell)$.*

Proof. If we look at the general description of the Renumbering Algorithm, we see that in each case only a constant number of steps are performed. These are either elementary operations on the graph, or dictionary lookups. There is however one important exception to this. In cases where we need to merge two topological edges, the renumbering of regular vertices (and simultaneously adjustment of their weights) is needed. Since every elementary operation on the graph takes constant time, and every dictionary lookup takes $O(\log \ell)$ time, all we have to prove is that the total number of renumberings is $O(\ell \log \ell)$.

A key concept in our proof is the notion of a *super edge* (see Figure 6.10). Super edges are sets of topological edges which can be defined inductively: initially each topological edge (with one or two regular vertices on it) is a member of a separate super edge. If a member \mathbf{a} of a super edge \mathbf{A} is merged with a member \mathbf{b} of another super edge \mathbf{B} , then the two super edges are unioned together in a new super edge \mathbf{C} and \mathbf{a} and \mathbf{b} are merged into a new member \mathbf{c} of the new super edge \mathbf{C} . If a member \mathbf{d} of a super edge is split into \mathbf{e} and \mathbf{f} , then both \mathbf{e} and \mathbf{f} will belong to the same super edge as \mathbf{d} did. The important property of super edges is that the total number of vertices can only grow. We call this number the *size* of a super edge. A split operation does not affect the size of super edges, while merge operations can only increase it.

We now prove by induction on ℓ that the total number of renumberings in a super edge of size ℓ is $O(\ell \log \ell)$.

The statement is trivial for $\ell = 0$, so we take $\ell > 0$. We may assume that the ℓ th update involves a merge of two topological edges, since this is the only update for which we have to do renumbering. Suppose that the sizes of the two super edges being unioned are ℓ_1 and ℓ_2 . Without loss of generality assume that $\ell_1 \leq \ell_2$. Hence, according to the Renumbering Algorithm which renumbers the shortest of the two, we have to do ℓ_1 renumbering steps: ℓ_1 to assign new numbers, and ℓ_1 to assign

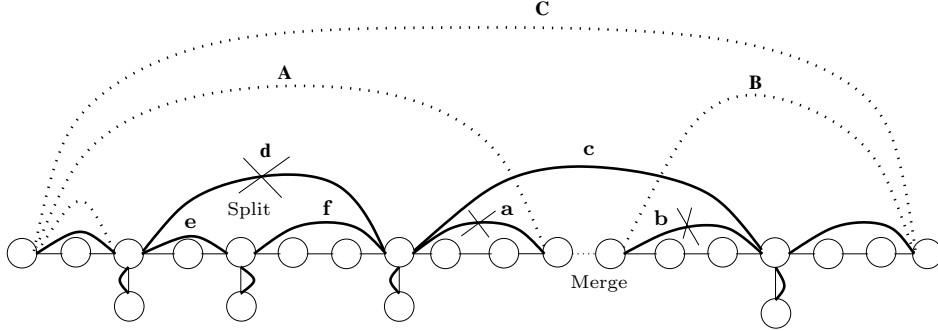


Figure 6.10: An example of some super edges (dotted lines)

new weights. The size of the new super edge will be $\ell = \ell_1 + \ell_2$. By the induction hypothesis, the total numbers of renumberings already done while building the two given super edges are $\ell_1 \log \ell_1$ and $\ell_2 \log \ell_2$. It is known that

$$2 \min\{x, 1-x\} \leq x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}, \quad (6.2.1)$$

for $x \in [0, 1]$ [67]. Define $x = \ell_1/\ell$. By (6.2.1), we then obtain the following inequality

$$\ell_1 \log \ell_1 + \ell_2 \log \ell_2 + 2\ell_1 \leq \ell \log \ell,$$

as had to be shown. \square

To conclude this section, we recall from the previous section that the maximal number assigned to a regular vertex is $2\ell^2$. So, all numbers involved in the Renumbering Algorithm take only $O(\log \ell)$ bits in memory. Indeed, Theorem 6.2.1 assumes the standard RAM computation model with unit costs. If logarithmic costs are desired, the complexity is $O(\ell \log^2 \ell)$.

6.2.4 The Topology Tree Algorithm

In this section we introduce another algorithm for keeping the topological invariant of a graph up-to-date when this graph is subject to edge insertions. We only describe the case of edge insertion, but it is straightforward to extend the Topology Tree Algorithm to a fully dynamic algorithm. The algorithm is based on the topology tree, which has been introduced by Frederickson [21, 22], and is used extensively in other partially, and fully dynamic algorithms [38].

We first show how the topological edge can be found efficiently.

Regular multilevel partition We define a *cluster* as a set of vertices. The *size* of a cluster is the number of vertices it contains. A *regular cluster* is a cluster of size at most two, containing adjacent regular vertices. A *regular partition* of a graph G is

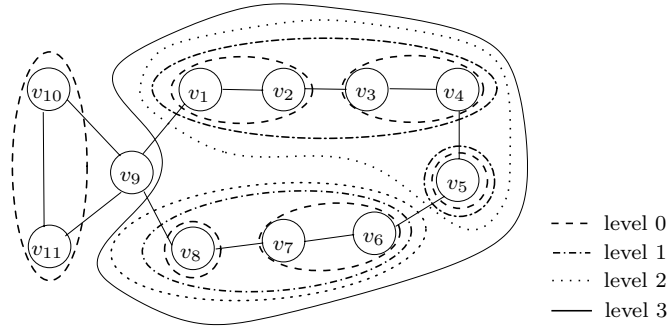


Figure 6.11: Example of a regular multilevel partition of a graph.

a partition of the set of regular vertices, $V_r = V - V_I$, such that for any two adjacent regular vertices v and w , the following holds:

- either v and w are in the same regular cluster \mathcal{C} ; or
- v and w are in different regular clusters \mathcal{C}_v and \mathcal{C}_w , and at least one of these regular clusters has size two.

A *regular multilevel partition* of a graph G is a set of partitions of V_r that satisfy the following (see Figure 6.11):

1. For each level $i = 0, 1, \dots, k$, the clusters at level i form a partition of V_r .
2. The clusters at level 0 form a regular partition of V_r .
3. The clusters at level i form a regular partition when viewing each cluster at level $i - 1$ as a regular vertex.

A *regular forest* of a graph G is a forest based on a regular multilevel partition of G .

We focus on the construction of a single tree in the forest corresponding to a single regular path. A single tree is constructed as follows (see Figure 6.12).

1. A vertex at level i in the tree represents a cluster at level i in the regular multilevel partition.
2. A vertex at level $i > 0$ has children that represent the clusters at level $i - 1$ whose union is the cluster it represents.

The height of a topology tree is logarithmic in the number of regular vertices in the leafs [21].

We also store adjacency information for the clusters. Two regular clusters \mathcal{C} and \mathcal{C}' at level 0 are *adjacent*, if there exists a vertex $v \in \mathcal{C}$ and a vertex $w \in \mathcal{C}'$ such that v and w are adjacent in G .

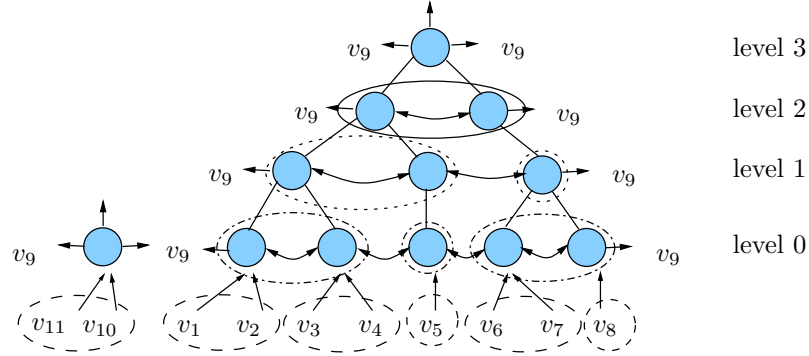


Figure 6.12: The regular forest corresponding to the regular multilevel partition shown in Figure 6.11

We call two clusters \mathcal{C} and \mathcal{C}' at level i adjacent, if they have adjacent children. A regular cluster \mathcal{C} at level 0 is adjacent to a singular vertex s if there exists a regular vertex $v \in \mathcal{C}$ adjacent to s . A cluster at level $i > 0$ is adjacent to a singular vertex s if it has a child adjacent to s .

Maintaining a regular multilevel partition It is very easy to adjust the regular partition, i.e., the regular clusters at level 0 of the regular multilevel partition. When an edge $e = \{x, y\}$ is inserted, we distinguish between the following cases: 1. the edge e destroys a regular vertex u ; 2. the edge e destroys two regular vertices u and v ; 3. the edge e creates a regular vertex u ; 4. the edges e creates two regular vertices u and v ; 5. the edge e does not change the number of regular vertices. We denote with C_u (C_v) the regular cluster containing the vertex u (v). We treat these cases as follows.

1. If the size of C_u is 1, then this cluster is deleted. Otherwise if C_u is adjacent to a cluster C of size one, remove u from C_u and union C_u with C .
2. Apply case 1 to both C_u and C_v .
3. Create a new cluster C_u only containing u . If C_u is adjacent to a cluster C of size one, union C_u with C .
4. Apply case 3, but if both C_u and C_v are not adjacent to a cluster of size one, then they are unioned together.
5. Nothing has to be done.

As an example consider the graph depicted in Figure 6.13. The insertion of edge $\{x, y\}$ destroys the regular vertex x , so we are in case 1. Because \mathcal{C}' is adjacent to \mathcal{C}'' and the size of \mathcal{C}'' is one, we must union \mathcal{C}' and \mathcal{C}'' together into a new regular cluster \mathcal{C} . The maintenance of the regular partition is completed after adjusting the adjacency information of both \mathcal{C} and \mathcal{D} , as shown in Figure 6.13.

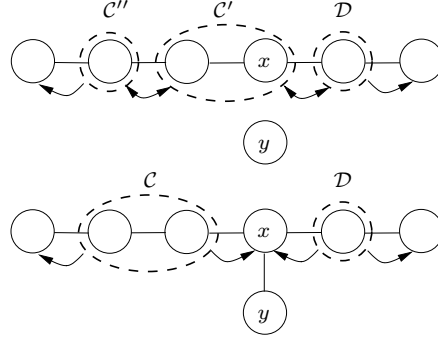


Figure 6.13: Adjusting the regular partition after inserting edge $\{x, y\}$.

We assume that the regular partition at level 0 reflects the insertion of an edge, as discussed above. The number of clusters which have changed, inserted or deleted is at most some constant. We put these clusters in a list L_C , L_I , and L_D according to whether they are changed, inserted or deleted. More specifically, these lists are initialized as follows. Each regular cluster that has been split or combined to form a new regular cluster is inserted in L_D , while each new regular cluster is inserted in list L_I . The adjacency information is stored with the clusters in L_I . For clusters in L_D every adjacency information is set to null, except the parent information. For each regular cluster whose set of vertices has not changed but its adjacency information has changed, update the adjacency information and insert it into L_C .

We create lists L'_D , L'_I , and L'_C to hold the clusters at the next higher level of the regular multilevel partition. These lists are initially empty.

We first adjust the clusters in the list L_D . Every cluster \mathcal{C} in L_D is removed from L_D , and \mathcal{C} is removed as child from its parent \mathcal{P} (if this exists).

- If \mathcal{P} has no children, then insert \mathcal{P} in L'_D .
- If \mathcal{P} still has a child \mathcal{C}' , then if \mathcal{C}' is not already in L_C or L_D , then insert \mathcal{C}' into L_C .

Next, we search the list L_C for clusters that have siblings. Suppose that $\mathcal{C} \in L_C$ has a sibling \mathcal{C}' and parent \mathcal{P} .

- If \mathcal{C} and \mathcal{C}' are adjacent, then remove \mathcal{C} from the list L_C , and remove \mathcal{C}' from L_C if it is in this list. Insert \mathcal{P} into L'_C .
- If \mathcal{C} and \mathcal{C}' are not adjacent, then remove \mathcal{C} and \mathcal{C}' as children from \mathcal{P} . Remove \mathcal{C} from the list L_C , and also remove \mathcal{C}' from L_C if it is in this list. Insert both \mathcal{C} and \mathcal{C}' into L_I , and insert \mathcal{P} in L'_D .

Finally, we treat the remaining clusters in L_C and in L_I . Let \mathcal{C} be such a cluster. Remove \mathcal{C} from the appropriate list. In what follows, the degree of \mathcal{C} is the number of adjacent clusters.

- If \mathcal{C} has degree zero, then it is the root of a tree in the regular forest. Insert its parent \mathcal{P} in L'_D (if it exists).
- If \mathcal{C} has degree one or two, then we have the following possibilities:
 - If every adjacent cluster to \mathcal{C} has a sibling, then insert the parent \mathcal{P} of \mathcal{C} into L'_C in case \mathcal{P} exists. In case \mathcal{C} does not have a parent, create a new parent cluster \mathcal{P} and insert it into L'_I .
 - Let \mathcal{C}' be a cluster adjacent to \mathcal{C} which has no sibling. Remove \mathcal{C}' from the appropriate list, if it is in a list. If both \mathcal{C} and \mathcal{C}' have a parent, denoted by \mathcal{P} and \mathcal{P}' respectively, then remove \mathcal{C} as child of \mathcal{P} and make it a child of \mathcal{P}' . Insert \mathcal{P} into L'_D , and insert \mathcal{P}' into L'_C . If both \mathcal{C} and \mathcal{C}' have no parent, then create a new parent \mathcal{P} of \mathcal{C} and \mathcal{C}' , and insert \mathcal{P} into L'_I . If \mathcal{C} has a parent \mathcal{P} , and \mathcal{C}' has no parent, then make \mathcal{C}' a child of \mathcal{P} and insert \mathcal{P} into L'_C . The case that \mathcal{C}' has a parent \mathcal{P}' , and \mathcal{C} has no parent is analogous.

When all clusters are removed from L_D , L_C , and L_I , determine and adjust the adjacency information for all clusters in L'_D , L'_C , and L'_I and reset L_C to be L'_C , L_D to be L'_D , and L_I to be L'_I . If no clusters are present in L'_D , L'_C or L'_I , nothing needs to be done and the iteration stops. This completes the description of how to handle the lists L_D , L_C , and L_I .

Finding a topological edge Consider that we are in one of the cases 4–6 described in Section 6.2.2, where we have to split a topological edge. Let x be the regular vertex at which we have to split the topological edge. We store a pointer from x to the regular cluster \mathcal{C}_x in which it is contained. We also store a pointer from each root of a tree T in the regular forest to the topological edge, corresponding to the regular path formed by all vertices in the leaves of T . We find the topological edge which needs to be split by going from \mathcal{C}_x to the root of the tree containing \mathcal{C}_x . Since the height of the tree is at most $O(\log \ell)$, where ℓ is the current number of edge insertions, we obtain the following.

Proposition 6.2.3. *Given a regular vertex x , the regular forest returns the topological edge corresponding to the regular path on which this regular vertex lies in $O(\log m)$ time.*

Storing weight information We store weight information in two different places. We define the weight of a regular cluster at level 0 of size one as zero. Let \mathcal{C} be a cluster at level 0 of size two, and let v and w be the two regular vertices in \mathcal{C} . Then we define the weight of \mathcal{C} as the weight of the edge $\{v, w\}$. If a cluster at level 0 is adjacent to a singular vertex s , then we store the weight of $\{v, s\}$ together with the adjacency information (Here, v is the vertex in \mathcal{C} adjacent to s). If two clusters \mathcal{C} and \mathcal{C}' at level 0 are adjacent, then we store the weight of $\{v, w\}$ together with their adjacency information (here $v \in \mathcal{C}$ and $w \in \mathcal{C}'$ and v is adjacent to w).

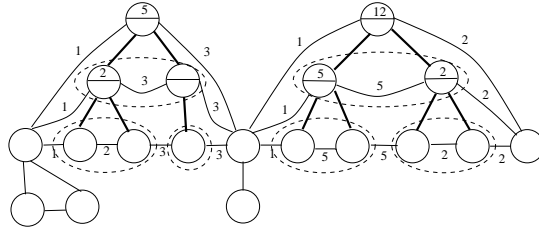


Figure 6.14: Example of a regular tree together with its weight information.

The weight of a cluster of size one at level $i > 0$, is defined as the weight of its child at the next lower level. The weight of a cluster of size two at level $i > 0$ equals the sum of the weights of its two children and the weight stored with their adjacency information. If two clusters at level $i > 0$ are adjacent, we store the weight of the adjacency information of their adjacent children. If a cluster at level $i > 0$ is adjacent to a singular node, we store the weight of the adjacency information of its child and the singular node.

Maintaining weight information The weight of clusters and the weights stored together with the adjacency information, is updated after each run of the update procedure for the regular multilevel partition, with an extra constant cost. Indeed, both the weights of clusters at level 0 and the weights stored with the adjacency information, are trivially updated. When we assume that all levels lower than i represent the weight information correctly, the weight information of clusters in L_C and L_I is trivially updated using the weight information at level $i - 1$.

Finding the weights As mentioned above, each root of a regular tree in the regular forest, has a pointer to a unique topological edge. This root has its own weight, as defined above, and is adjacent to two singular vertices. The weight of the topological edge is obtained by summing the weight of the root together with the weights of the adjacency information of the two singular vertices. This is illustrated in Figure 6.14

Complexity Analysis The complexity of the Topology Tree Algorithm is governed by two things: the maximal height of a single tree in the regular forest, and the amount of work that needs to be done at each level in the maintenance of the regular multilevel partition. The height of a single tree is easily shown to be logarithmic in the number of regular vertices on the regular path on which the tree is built. Moreover, it is shown by Frederickson [21], that in the lists L_C , L_D , and L_I only a constant number of clusters are stored. This means that since these lists are updated at most $O(\log \ell)$ times, where ℓ is the number of edge insertion, the total update time is still $O(\log \ell)$ per edge insertion. Hence, we may conclude that

Theorem 6.2.2. *The total time spent on ℓ updates by the Topology Tree Algorithm is $O(\ell \log \ell)$.*

6.2.5 Experimental Results

For our experiments, we used the implementations of the Renumbering Algorithm and Topology Tree Algorithm. The correctness of the implementations was checked by comparing the topological invariants of graphs obtained by a random sequence of edge insertions.

Test Environment

Both implementations were implemented in C++ using LEDA. We used the GNU g++ compiler version 2.95.2 without any optimization option. Our experiments were performed on a SUN Ultra 10 running at 440 Mhz with 512Mb internal memory. We measured the time for performing the sequence of updates.

A comparative test

We conducted our experiments on three types of inputs. First of all, we extensively studied random inputs, which are random sequences of updates on random graphs. This establishes the average case performance of the algorithms. Next, we used two kinds of non-random graph inputs which focus on specific features of the algorithms. More specifically, we constructed a graph input which repeatedly merges topological edges and a graph input which first creates a very large number of small topological edges, and then splits these edges randomly. Finally, we ran both algorithms on two graph instances originating from real data sets.

Random Inputs The random inputs consist of random graphs that are generated, given the number of vertices and edges. Each experiment builds the random graph incrementally with the insertions uniformly distributed over the set of edges. We conducted a series of tests for different number of nodes n and number of edges m . For every pair of values for n and m we did 1000 experiments and computed the ratio between the total time the Topology Tree Algorithm needed to perform the test and the total time the Renumbering Algorithm needed to accomplish the same task. We took the average of this ratio and computed the 95% confidence interval. A value smaller than 1 means that the Topology Tree Algorithm is faster in 95% of the cases. The results of these experiments are shown in Table 6.1.

For small numbers of edge insertions, i.e., when the probability of having many regular vertices is large, we see that the Renumbering Algorithm is faster. However, when the number of edge insertions increases, the Topology Tree Algorithm becomes slightly faster. This is probably due to the fact that the dictionary in the Renumbering Algorithm becomes very large, i.e., there are many short topological edges, and hence, it takes longer to search a topological edge. The Topology Tree Algorithm probably becomes faster because the heights of the trees in the regular forest become smaller. This is confirmed by the results of experiments on non-random inputs.

Non-Random Inputs The non-random inputs consisted of two types. For the first type, we first created a large number of topological edges and then started to

vertices\edges	m=5 000	m=10 000	m=20 000
n=1 000	[1.10, 1.15]	[1.03, 1.06]	[0.97, 0.99]
vertices\edges	m=5 000	m=25 000	m=75 000
n=5000	[1.25, 1.29]	[1.01, 1.03]	[0.96, 0.98]
vertices\edges	m=10 000	m=50 000	m=150 000
n=50 000	[1.30, 1.35]	[1.06, 1.07]	[0.91, 0.92]
vertices\edges	m=10 000	m=100 000	m=300 000
n=100 000	[1.21, 1.23]	[0.98, 0.99]	[0.85, 0.86]

Table 6.1: Experimental results on random inputs.

vertices\edges	m=20 000
n=10 000	[1.7, 1.75]
vertices\edges	m=60 000
n=15 000	[0.85, 0.87]

Table 6.2: Experimental results on two types of non-random inputs.

merge these edges pairwise. The end result was a very long topological edge. For the second type, we first created a very large number of topological edges corresponding to a regular path consisting of a single regular vertex, and then started to split the topological edges randomly in the regular vertices. The results shown in Table 6.2 are obtained after doing 200 experiments. In each experiment, we computed the ratio between the total time the Topology Tree Algorithm needed to perform the test and the total time the Renumbering Algorithm needed to accomplish the same task. We took the average of this ratio and computed the 95% confidence interval. For the first type of inputs, the Topology Tree Algorithm has to maintain large topology trees, which is probably the reason that it is slower than the Renumbering Algorithm. For the second type of inputs, the topology trees all have height one, but the dictionary is very large. This is probably the reason that the Renumbering Algorithm is slower on these kinds of inputs.

Real Data Inputs We also tested the relative performance of both algorithms with respect to graphs representing real data. We present the results on two data sets:

Hydrography graph A data set representing the hydrography of Nebraska. This set contains 157 972 vertices, of which 96 636 are regular.

Railroad graph A data set representing all railway mainlines, railroad yards, and major sidings in the continental U.S. compiled at a scale 1 : 100 000. It contains 133 752 vertices of which only 14 261 are regular.

The results shown in Table 6.3 are obtained after performing 200 experiments. In each experiment, we ran both algorithms in a random way on these data sets. We

vertices\edges	m=101 336
n=157 972	[1.60, 1.63]
vertices\edges	m=164 380
n=133 752	[0.94, 0.95]

Table 6.3: Experimental results on two real data sets.

computed the ratio between the total time the Topology Tree Algorithm needed to perform the test and the total time the Renumbering Algorithm needed to accomplish the same task. We took the average of this ratio and computed the 95% confidence interval. Again, we see that when there are only few, but long, topological edges, the Renumbering Algorithm is faster than the Topology Tree Algorithm. When there are many, short, topological edges, like in the railroad graph, the Topology Tree Algorithm is slightly faster than the Renumbering Algorithm.

Bibliography

- [1] D. Abel and B.C. Ooi, editors. *Advances in Spatial Databases—3rd Symposium SSD'93*, volume 692 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
- [2] D. Alberts, G. Cattaneo, and G.F. Italiano. An empirical study of dynamic graph algorithms. *ACM Journal of Experimental Algorithms*, 2(5), 1997.
- [3] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1046, 1996.
- [4] R. Benedetti and J.J. Risler. *Real Algebraic and Semi-algebraic Sets*. Hermann, Paris, 1990.
- [5] M. Benedikt, M. Grohe, L. Libkin, and L. Segoufin. Reachability and connectivity queries in constraint databases. In *Proceedings of the 19th ACM Symposium on Principles of Database Systems*, pages 104–115. ACM Press, 2000.
- [6] M. Benedikt and L. Libkin. Safe constraint queries. *Siam Journal on Computing*, 29(5):1652–1682, 2000.
- [7] M. Benedikt and L. Libkin. Exact and approximate aggregation in constraint query languages. In *Proceedings of the 18th ACM Symposium on Principles of Database Systems*, pages 102–113. ACM Press, 1999. To appear in the *Journal of Computer and System Sciences*.
- [8] J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry*, volume 36 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, 1998.
- [9] A. P. Buchmann, O. Günther, T. R. Smith, and Y.-F. Wang, editors. *Design and implementation of large spatial databases—1st Symposium*, volume 409 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.
- [10] A. Chandra and D. Harel. Computable queries for relational data bases. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.
- [11] B. Clark. A calculus of individuals based on “connection”. *Notre Dame Journal of Formal Logic*, 22(3):204–218, 1981.

-
- [12] A. G. Cohn, D. A. Randell, and Z. Cui. Taxonomies of logically defined qualitative spatial relations. In N. Guarino and R. Poli, editors, *Formal ontology in conceptual analysis and knowledge representation*, 43(5-6):831–846, Kluwer, 1995.
- [13] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, Berlin, 1975. Springer-Verlag.
- [14] T.H. Cormen, C.E. Leieron, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [15] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry—Algorithms and Applications*. Springer-Verlag, 1997.
- [16] F. Dumortier, M. Gyssens, L. Vandeurzen, D. Van Gucht. On the Decidability of Semilinearity for Semialgebraic Sets and Its Implications for Spatial Databases. *Journal of Computer and System Sciences*, 58(3):535–571, 1999.
- [17] H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, Berlin, 1995.
- [18] M. Egenhofer. A model for detailed binary topological relationships. *Geomatica*, 47(3–4):261–273, 1993.
- [19] M.J. Egenhofer and J.R. Herring, editors. *Advances in Spatial Databases—4th Symposium SSD’95*, volume 951 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
- [20] D. Eppstein, Z. Galil, G.F. Italiano, and A. Nissenzweig. Sparsification - a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44(5):669–696, 1997.
- [21] G.N. Frederickson. Data structures for on-line updating of minimal spanning trees. *SIAM Journal of Computing*, 14:781–798, 1985.
- [22] G.N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. *SIAM Journal of Computing*, 26(2):484–538, 1997.
- [23] F. Geerts and B. Kuijpers. Expressing topological connectivity of spatial databases. In *Research Issues in Structured and Semistructured Database Programming. Proceedings of 8th International Workshop on Database Programming Languages*, volume 1949 of *Lecture Notes in Computer Science*, pages 224–238. Springer-Verlag, 1999.
- [24] F. Geerts and B. Kuijpers. Linear approximation of planar spatial databases using transitive-closure logic. In *Proceedings of the 19th ACM Symposium on Principles of Database Systems*, pages 126–135. ACM Press, 2000.

-
- [25] F. Geerts, B. Kuijpers, and J. Van den Bussche. Topological canonization of planar spatial data and its incremental maintenance. In T. Polle, T. Ripke, and K.-D. Schewe, editors, *Proceedings of the 7th International Workshop on Foundations of Models and Languages for Data and Objects*, pages 55–68. Kluwer Academic Publisher, 1998.
- [26] C. G. Gibson, K. Wirthmüller, A. A. du Plessis, and E. J. N. Looijenga. *Topological Stability of Smooth Mappings*, volume 552 of *Lecture Notes in Mathematics*. Springer-Verlag, 1976.
- [27] M. Grohe and L. Segoufin. On first-order topological queries. In *Proceedings of the 15th IEEE Symposium on Logic in Computer Science*, pages 349–360. IEEE Press, 2000. To appear in the ACM Transactions on Computational Logic.
- [28] S. Grumbach and G. Kuper. Tractable recursion over geometric data. In G. Smolka, editor, *Proceedings of the 3rd Conference on Principles and Practice of Constraint Programming*, number 1330 in *Lecture Notes in Computer Science*, pages 450–462. Springer-Verlag, 1997.
- [29] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. DEDALE, a spatial constraint database. In S. Cluet and R. Hull, editors, *Proceedings of the 7th Workshop on Database Programming Languages*, volume 1369 of *Lecture Notes in Computer Science*, pages 38–59. Springer-Verlag, 1998.
- [30] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In L.M. Haas and A. Tiwary, editors, *Proceedings of the ACM International Conference on Management of Data*, pages 213–224. ACM Press, 1998.
- [31] S. Grumbach and J. Su. Towards practical constraint databases. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems*, pages 28–39. ACM Press, 1996.
- [32] S. Grumbach and J. Su. Queries with arithmetical constraints. *Theoretical Computer Science*, 173(1):151–181, 1997.
- [33] V. Guillemin and A. Pollack. *Differential topology*. Prentice-Hall, 1974.
- [34] O. Gunther and H.-J. Schek, editors. *Advances in Spatial Databases—2nd Symposium SSD’91*, volume 525 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.
- [35] R.H. Güting, editor. *Advances in Spatial Databases—6th Symposium SSD’99*, volume 1651 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
- [36] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometrical query languages. *Journal of Computer and System Sciences*, 58(1):483–511, 1999.

-
- [37] J. Heintz, T. Recio, and M.-F. Roy. Algorithms in real algebraic geometry and applications to computational geometry. In J. Goodman, R. Pollack, and W. Steiger, editors, *Discrete and Computational Geometry*, volume 6. AMS-ACM, 1991.
- [38] G. Italiano. Dynamic graph algorithms. In Mikhail J. Atallah, editor, *Handbook on Algorithms and Theory of Computation*. CRC Press, 1998.
- [39] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Science*, 51(1):26–52, 1995.
- [40] P. Koiran. Approximating the volume of definable sets. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 134–141. IEEE Press, 1995.
- [41] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132:113–128, 1994.
- [42] D. Kozen and C.-K. Yap. Algebraic cell decomposition in nc. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 515–521. IEEE Press, 1985.
- [43] D.C. Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, 1992.
- [44] S. Kreutzer. Operational semantics for fixed-point logics on constraint databases. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science. Springer-Verlag, 2001.
- [45] S. Kreutzer. Query languages for constraint databases: First-order logic, fixed-points, and convex hulls. In J. Van den Bussche and V. Vianu, editors, *Proceedings of the 9th International Conference on Database Theory*, volume 1973 of *Lecture Notes in Computer Science*, pages 248–262. Springer-Verlag, 2001.
- [46] B. Kuijpers. *Topological Properties of Spatial Databases in the Polynomial Constraint Model*. PhD thesis, University of Antwerp (UIA), 1998.
- [47] B. Kuijpers and J. Paredaens and J. Van den Bussche.. Lossless representation of topological spatial data In Egenhofer and Herring [19], pages 1–13.
- [48] B. Kuijpers, J. Paredaens, M. Smits, and J. Van den Bussche. Termination properties of spatial datalog. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases*, volume 1154 of *Lecture Notes in Computer Science*, pages 101–116. Springer-Verlag, 1996.
- [49] B. Kuijpers, J. Paredaens, and J. Van den Bussche. Topological elementary equivalence of closed semi-algebraic sets in the real plane. *Journal of Symbolic Logic*, 65(4):1530–1555, 2000.

-
- [50] B. Kuijpers and M. Smits. On expressing topological connectivity in spatial datalog. In V. Gaede, A. Brodsky, O. Gunter, D. Srivastava, V. Vianu, and M. Wallace, editors, *Proceedings of the 2nd Workshop on Constraint Databases and Applications*, volume 1191 of *Lecture Notes in Computer Science*, pages 116–133. Springer-Verlag, 1997.
- [51] G.M. Kuper, J. Paredaens, and L. Libkin, editors. *Constraint Databases*. Springer-Verlag, 2000.
- [52] John. M. Lee. *Introduction to Topological Manifolds*, volume 202 of *Graduate Texts in Mathematics*. Springer-Verlag, 2000.
- [53] K. Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*. EACTS Monographs on Theoretical Computer Science. Springer-Verlag, 1984.
- [54] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.
- [55] E.E. Moise. *Geometrical Topology in Dimensions 2 and 3*. Springer-Verlag, 1977.
- [56] J. O’Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.
- [57] C. H. Papadimitriou, D. Suciú, and V. Vianu. Topological queries in spatial databases. *Journal of Computer and System Sciences*, 58(1):29–53, 1999.
- [58] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems*, pages 279–288. ACM Press, 1994.
- [59] Ian Pratt and Dominik Schoop. Expressivity in polygonal, plane mereotopology. *Journal of Symbolic Logic*, 65(2):822–838, 2000.
- [60] F.P. Preparata and M.I. Shamos. *Computational Geometry—An Introduction*. Springer-Verlag, 1985.
- [61] E. Rannou. The complexity of stratification computation. *Discrete and Computational Geometry*, 19:47–79, 1998.
- [62] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13:1992, 1992.
- [63] M. Scholl and A. Voisard, editors. *Advances in Spatial Databases—5th Symposium SSD’97*, volume 1262 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
- [64] L. Segoufin and V. Vianu. Querying spatial databases via topological invariants. *Journal of Computer and System Sciences*, 61(2):270–301, 2000.
- [65] A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

-
- [66] M. Shiota. *Geometry of Subanalytic and Semialgebraic Sets*. Birkhäuser, 1997.
- [67] R. Tamassia. On-line planar graph embedding. *Journal of Algorithms*, 21:201–239, 1996.
- [68] R.E. Tarjan. Data structures and network algorithms. In *CBMS-NSF Regional Conference Series in Applied Mathematics*, volume 44. SIAM, 1983.
- [69] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.
- [70] D. Thompson and R. Laurini. *Fundamentals of Spatial Information Systems*. Number 37 in APIC Series. Academic Press, 1992.
- [71] L. van den Dries. *Tame Topology and O-minimal Structures*. Cambridge University Press, 1998.
- [72] L. van den Dries and C. Miller. Geometric categories and O-minimal structures. *Duke Mathematical Journal*, 82(2):497–540, 1996.
- [73] L. Vandeurzen. *Logic-Based Query Languages for the Linear Constraint Database Model*. PhD thesis, Limburgs Universitair Centrum (LUC), 1999.
- [74] L. Vandeurzen, M. Gyssens, and D. Van Gucht. On the desirability and limitations of linear spatial query languages. In Egenhofer and Herring [19], pages 14–28.
- [75] L. Vandeurzen, M. Gyssens, and D. Van Gucht. An expressive language for linear spatial database queries. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, pages 109–118. ACM Press, 1998.
- [76] H. Whitney. Local properties of analytical varieties. *Differential And Combinatorial Topology*, pages 205–244, 1965.
- [77] H. Whitney. Tangents to an analytic variety. *Annals of Mathematics*, 81:496–549, 1965.
- [78] A.J. Wilkie. On defining C^∞ . *Journal of Symbolic Logic*, 59:344, 1994.
- [79] A.J. Wilkie. A theorem of the complement and some new o-minimal structures. *Selecta Mathematica, New Series*, 5:397–421, 1999.
- [80] M. F. Worboys. *GIS: A Computing Perspective*. Taylor&Francis, 1995.
- [81] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1998.

Index

- ε -neighborhood, 47
- ε -approximation
 - algebraic, 27
 - query, 28
 - rational, 27
- FO+LIN, 13
- FO+POLY, 12
- algorithm
 - fully dynamic, 84
- almost all, 40
- amortized complexity, 93
- arity, 12
- box, 41
 - center, 41
 - collection, 49
 - covering, 52
 - covering query, 53
 - diameter, 41
 - dimension, 41
- cluster, 94
 - adjacent, 95
 - regular, 94
 - size, 94
- coalesce class, 82
- cone radius, 41
 - at infinity, 68
 - query, 41
 - uniform, 46
- connectivity
 - query, 14
- constraint database
 - A**-linear, 13
 - Z**-linear, 13
 - linear, 13
 - schema, 10
- critical
 - points, 42
 - value, 42
- decomposition
 - compatible, 38
 - regular, 36
 - uniform cone radius, 47
 - Whitney, 37
- diameter
 - of a box, 41
 - of a set, 72
- dictionary, 90
- dimension, 29
- doubly-connected edge list, 80
- Euclidean norm, 27
- extremal points, 73
- flow of vector field, 44
- general position, 39
- geometric realization, 41
- gradient vector field, 43
- graph
 - concrete, 78
 - equivalent, 79
 - labeled, 78
 - plane, 78
- homeomorphism, 14
- isotopic, 79
- line

- next, 80
 - twin, 78
- linear constraint
 - formula
 - \mathbf{Z} -linear, 13
- linearization
 - algebraic, 27
 - query, 28
 - rational, 27
- measure zero, 40
- polynomial constraint
 - calculus, 12
 - database
 - semantic, 10
 - syntactic, 10
 - formula, 8
- quantifier elimination, 9
- query, 11
 - Boolean, 12
 - evaluation, 12
 - expressible in, 12
 - genericity, 14
 - topological, 14
 - volume approximation, 71
- regular
 - cluster, 94
 - decomposition, 36
 - forest, 95
 - in a point, 34
 - multilevel partition, 95
 - partition, 95
 - path, 87
 - set, 34
- regular vertex, 87
 - number, 90
 - weight, 92
- semi-algebraic set, 8
- semi-linear set, 13
- singular vertex, 87
- stable property, 40
- stop condition, 17
- tangent space, 34
 - query, 34
- topological
 - edges, 87
- topological invariant
 - of a graph, 87
 - of planar graph, 79
 - of semi-algebraic set, 80
- topological type, 41
- topologically equivalent, 14
- transitive closure logic
 - FO+LIN+TC, 16
 - FO+LIN+TCS, 17
 - FO+POLY+TC, 15
 - FO+POLY+TCS, 16
- transversal intersection, 39
- triangulation, 74
- uniform cone radius
 - decomposition, 47
- volume, 71
 - approximtion query, 71
- Whitney
 - decomposition, 37
 - compatible, 38
 - property, 36
 - umbrella, 36

Samenvatting

Gedurende de laatste jaren is de behoefte aan gegevensbanksystemen die, naast de klassieke alfanumerieke gegevens, ook overweg kunnen met ruimtelijke gegevens, zeer sterk toegenomen. Eén van de gegevensbankmodellen die voorgesteld werden om dit probleem op te lossen, is het polynomiale constraint model.

Het polynomiale constraint model is een uitbreiding van het welbekende relationele model, het standaard model voor gegevensbanken die alleen maar alfanumerieke gegevens verwerken. In het polynomiale constraint model worden de mogelijke oneindige verzamelingen punten in de n -dimensionale ruimte \mathbf{R}^n eindig voorgesteld door Booleaanse combinaties van polynomiale vergelijkingen en ongelijkheden. De ruimtelijke gegevens die zo gerepresenteerd kunnen worden, noemen we semi-algebraïsche verzamelingen. Dergelijke gegevensbanken noemen we polynomiaal. Indien er slechts lineaire vergelijkingen en ongelijkheden met gehele coëfficiënten voorkomen in de representatie, spreken we over lineaire gegevensbanken.

De standaard bevragingstaal in het polynomiale constraint model is de relationale calculus, uitgebreid met polynomiale vergelijkingen en ongelijkheden. Uitdrukkingen in deze taal kunnen effectief berekend worden dankzij een kwantor-eliminatie-procedure, waarvan het bestaan werd aangetoond door Tarski.

Dit proefschrift handelt hoofdzakelijk over uitbreidingen van de zojuist genoemde calculus met recursie, en over de uitdrukingskracht van deze uitbreidingen met betrekking tot bevragingen over topologische eigenschappen van gegevensbanken.

We beginnen met de uitbreiding van de calculus met een recursiemechanisme, namelijk de transitieve sluiting. De transitieve sluiting is een heel eenvoudige vorm van recursie en we laten slechts een elementair gebruik van deze operator toe in deze recursieve bevragingstaal, die we FO+TC noemen. Wanneer we aan FO+TC een expliciete stop-conditie toevoegen, tonen we aan dat alle berekenbare bevragingen aan lineaire gegevensbanken gesteld kunnen worden. Vervolgens tonen we aan dat in FO+TC, uitgebreid met een expliciete stopconditie, alle berekenbare Booleaanse topologische bevragingen aan polynomiale gegevensbanken gesteld kunnen worden. We tonen dit aan door een linearisatie te construeren in FO+TC. Deze linearisatie is een lineaire gegevensbank die topologisch equivalent is met een gegeven, polynomiale gegevensbank. Op deze manier kunnen bevragingen, wat betreft de topologische eigenschappen, vertaald worden naar lineaire gegevensbanken toe. De constructie van de linearisatie steunt op de eigenschap van semi-algebraïsche verzamelingen dat er in

elk punt een zogenaamde “conische straal” bestaat. Binnen deze straal is de semi-algebraïsche verzameling topologisch equivalent met een conische figuur. We bewijzen een variant van deze eigenschap en tonen aan dat de conische straal een eigenschap is die reeds uitdrukbaar is in de ruimtelijke calculus. Dat deze linearisatie uitdrukbaar is in FO+TC heeft tot gevolg dat de topologische samenhang van polynomiale gegevensbanken uitgedrukt kan worden in deze taal. Een ander gevolg is dat deze linearisatie willekeurig dicht bij de polynomiale gegevensbank gekozen kan worden, zodat het tevens een benadering van de polynomiale gegevensbank wordt met betrekking tot metrische eigenschappen. We passen deze eigenschap toe op het benaderen van het volume van polynomiale gegevensbanken tot op willekeurige nauwkeurigheid.

We ronden dit proefschrift af met de studie van algoritmische aspecten van de zogenaamde topologische invariant van vlakke polynomiale gegevensbanken. Deze topologische invariant is een klassieke relationele gegevensbank die de topologische informatie van een vlakke polynomiale gegevensbank volledig bevat. Wanneer men nu topologische bevestigingen wil stellen aan de topologische invariant in plaats van aan de polynomiale gegevensbank, dan stelt zich het volgende probleem: hoe kan men op een efficiënte manier de topologische invariant aanpassen wanneer de polynomiale gegevensbank onderhevig is aan veranderingen? Eerst beschrijven we een algoritme voor deze taak, gebruik makend van een welbekende datastructuur om vlakke figuren voor te stellen. Dit algoritme reageert correct op elke mogelijke update. Vervolgens beschouwen we algemene grafen en geven twee algoritmen die de topologische invariant onderhouden wanneer enkel toevoegingen van bogen aan de grafen toegelaten zijn. Tenslotte bespreken we de relatieve performantie van de laatste twee algoritmen en geven aan wanneer welk van deze twee algoritmen het best gebruikt kan worden.