According to the guidelines of the Limburgs Universitair Centrum, a copy of this publication has been filed in the Royal Belgian Library I, Brussels, as publication D/1999/2451/20.

Logic-Based Query Languages for the Linear Constraint Database Model

Proefschrift voorgelegd tot het behalen van de graad van
doctor in de wetenschappen, richting informatica
aan het Limburgs Universitair Centrum
te verdedigen door

Luc Vandeurzen

Promotor:
Prof. Dr. M. Gyssens

Co-promotor:
Prof. Dr. D. Van Gucht

1999

Een woord van dank ...

Eerst en vooral wens ik mijn promotor Marc Gyssens en mijn co-promotor Dirk Van Gucht te bedanken voor al de tijd en moeite die zij gestoken hebben in het begeleiden van deze thesis. Ook wil ik Freddy Dumortier bedanken voor zijn "wiskundige" inbreng: zonder zijn hulp zou een deel van dit werk er wellicht helemaal anders hebben uitgezien. Jan Van den bussche verdient ook een pluim voor al die keren dat ik bij hem terecht kon met mijn vragen. Verder wil ik Jan Paredaens, Bart Kuijpers en Gabi Kuper bedanken voor de samenwerking.

Tenslotte nog een bedankje voor alle mensen die op een of andere manier hebben bijgedragen aan de realisatie van dit werk: ik denk in het bijzonder aan vrienden, familie, collega's en administratief en technisch personeel.

Samenvatting

De laatste jaren is er steeds meer behoefte aan gegevensbanksystemen die overweg kunnen met andere en vaak meer complexere gegevens dan enkel alfanumerieke gegevens die klassieke gegevensbanksystemen verwerken. Zo is er onder meer een toenemende vraag naar gegevensbanksystemen die ook *geometrische gegevens* kunnen opslaan en manipuleren. Eén van de gegevensbankmodellen die voorgesteld zijn om dit probleem op te lossen, is het *lineaire model*. Deze thesis draait volledig rond de studie van *bevragingstalen* voor het lineaire model en hun expressiviteit.

Het lineaire model is ontstaan als een restrictie van het *polynomiale model*. Het polynomiale model op zijn beurt is een uitbreiding van het welbekende *relationele model*, *het* standaard model voor gegevensbanken die alfanumerieke gegevens manipuleren. In het polynomiale model worden geometrische gegevens gerepresenteerd door eerste-orde logica formules samengesteld uit polynomiale vergelijkingen en ongelijkheden met gehele coëfficiënten of, equivalent, reële algebraïsche coëfficiënten, d.w.z., eerste-orde formules over de structuur $\langle \mathbf{R}, \leq, \times, +, 0, 1 \rangle$. De geometrische gegevens die zo gerepresenteerd kunnen worden, noemen we semi-algebraïsche verzamelingen. De standaard bevragingstaal in het polynomiale model is de relationele calculus uitgebreid met polynomiale vergelijkingen en ongelijkheden met gehele coëfficiënten of, equivalent, reële algebraïsche coëfficiënten. Deze taal zullen we voortaan FO + poly noemen. We kunnen nu op twee manieren het polynomiale model restricteren tot een lineair model. Vooreerst kunnen we ons beperken tot eerste-orde formules over de structuur $\langle \mathbf{R}, \leq, +, 0, 1 \rangle$. Het zo bekomen model zullen we het $\mathbf{Z}$-lineaire model noemen omdat de coëfficiënten van de lineaire vergelijkingen en ongelijkheden in dit model enkel gehele getallen, of equivalent, rationale getallen kunnen zijn. De geometrische gegevens die we zo kunnen representeren, noemen we $\mathbf{Z}$-semi-lineaire verzamelingen en de corresponderende beperking van de bevragingstaal FO + poly noemen we FO + $\mathbf{Z}$-linear. We kunnen ook minder streng optreden en lineaire vergelijkingen en ongelijkheden met algebraïsche coëfficiënten toelaten. Het re-

sulterende model noemen we het **A**-lineaire model, de geometrische gegevens die we kunnen voorstellen noemen we **A**-semi-lineaire verzamelingen en de corresponderende beperking van de bevragingstaal FO + poly noemen we FO + **A**-linear. Voor vele resultaten doet het er niet toe in welk model we werken en spreken we over het lineaire model, semi-lineaire verzamelingen en de bevragingstaal FO + linear. Er zijn echter resultaten waarbij het onderscheid tussen beide modellen wel belangrijk is. Voor die resultaten zullen we dat expliciet vermelden.

Wanneer we een taal gebruiken voor het ondervragen van een zekere klasse van gegevensbanken, is het erg belangrijk dat deze bevragingstaal *compatibel* is met die klasse van gegevensbanken. Hiermee bedoelen we dat iedere vraag uitgedrukt in deze taal steeds een gegevensbank van een zekere klasse zal afbeelden op een gegevensbank van dezelfde klasse. Een bevragingstaal die compatibel is met lineaire gegevensbanken zullen we kortweg een *lineaire* bevragingstaal noemen. Het is duidelijk niet zo dat FO+poly een lineaire bevragingstaal is en bijgevolg is FO+poly ongeschikt voor het ondervragen van lineaire gegevensbanken. We zullen een vraag *lineair* noemen wanneer ze steeds een lineaire gegevensbank afbeeldt op een lineaire gegevensbank. In deze thesis interesseren we ons in het bijzonder voor de klasse van lineaire vragen die uitdrukbaar zijn in FO + poly. En daarmee komen we tot een ander belangrijk begrip, en dat is *volledigheid*. We zeggen dat een bevragingstaal volledig is voor een bepaalde klasse van vragen wanneer die taal iedere vraag van die klasse kan uitdrukken.

We beginnen met de expressiviteit van de bevragingstaal FO + linear op lineaire gegevensbanken te onderzoeken. We laten zien dat een hele reeks topologische en geometrische eigenschappen, waaronder de dimensie, de reguliere punten en de lagen van de reguliere stratificatie van een semi-lineaire verzameling, loodrechte stand en evenwijdigheid, uitdrukbaar zijn in FO + linear. De uitdrukbaarheid van deze eigenschappen vormt de basis van vele resultaten in de rest van deze thesis.

We laten echter ook zien dat lang niet iedere lineaire vraag die uitdrukbaar is in FO + poly ook uitdrukbaar is in FO + linear. We ontwikkelen daarvoor een techniek die een verband legt tussen het niet uitdrukbaar zijn van een lineaire vraag in FO + linear en het niet semi-lineair zijn van een nauw verwante semi-algebraïsche verzameling. Dit laatste is vaak eenvoudiger te bewijzen. (Later tonen we zelfs aan dat het beslisbaar is of een semi-algebraïsche verzameling semi-lineair is of niet.) We gebruiken deze techniek om aan te tonen dat een aantal fundamentele vragen over semi-lineaire verzamelingen, ondermeer vragen betreffende Euclidische afstand, convex omhullende en Voronoi diagram, niet kunnen worden uitgedrukt in FO + linear. We besluiten dan ook dat FO + linear tekort schiet als algemene bevragingstaal voor lineaire gegevensbanken. Hiermee is ook de noodzaak verklaard voor het zoeken naar uitbreidingen van FO + linear die strikt expressiever zijn en toch compatibel blijven met lineaire gegevensbanken. Dergelijke uitbreidingen zullen we later in deze thesis

bestuderen.

Alhoewel FO + linear als algemene lineaire bevragingstaal geen goede keuze is, blijft het interessant om te weten welke lineaire vragen uitdrukbaar in FO + poly ook uitdrukbaar zijn in FO + linear. We tonen echter aan dat het in het algemeen niet beslisbaar is of een lineaire vraag die uitdrukbaar is in FO + poly ook uitdrukbaar is in FO + linear. We laten zelfs zien dat voor deelklassen van de lineaire vragen die uitdrukbaar zijn in FO + poly, zoals bijvoorbeeld de lineaire vragen die compatibel zijn met eindige gegevensbanken of de lineaire vragen die compatibel zijn met eindige unies van affiene ruimten, het in het algemeen onbeslisbaar blijft of vragen van die deelklasse uitdrukbaar zijn in FO + linear.

Vervolgens concentreren we ons op de vraag of we kunnen beslissen of een semi-algebraïsche verzameling semi-lineair is. We tonen aan dat dit inderdaad beslisbaar is, zowel in het **Z**-lineaire als het **A**-lineaire geval. We bewijzen echter ook dat er een FO + poly formule bestaat die **A**-lineariteit beslist, dit in tegenstelling met **Z**-lineariteit waarvan we laten zien dat die niet beslist kan worden met een FO + poly formule. Uit de gebruikte bewijstechnieken volgen bovendien een aantal belangrijke neven resultaten, zoals een algoritme om een willekeurige semi-lineaire verzameling op te splitsen in convexe verzamelingen en een algoritme om de "speciale punten" van een semi-lineaire verzameling te vinden.

We verleggen nu terug onze aandacht naar lineaire bevragingstalen, en meer in het bijzonder gaan we op zoek naar talen die strikt expressiever zijn dan FO + linear, zonder daarbij de compatibiliteit van deze talen op het spel te zetten.

We bestuderen eerst een techniek om FO + linear op een veilige manier uit te breiden met lineaire operatoren. Gezien de beperkte expressiviteit van FO + linear, zullen we op zijn minst lineaire operatoren moeten toevoegen die ons toelaten om onder meer met afstand en convex omhullende te kunnen werken om tot een praktisch bruikbare bevragingstaal voor lineaire gegevensbanken te komen.

Daarna experimenteren we met het toevoegen van een beperkte vorm van vermenigvuldiging aan FO + linear. De resulterende taal noemen we PFOL. We laten zien dat deze taal inderdaad een lineaire taal is. Vervolgens onderzoeken we de expressiviteit van deze nieuwe lineaire taal. Hiervoor ontwikkelen we eerst een coderingsalgoritme voor het vinden van een eindige representatie van een willekeurige semi-lineaire verzameling en een decoderingsalgoritme voor het herberekenen van een semi-lineaire verzameling uit zijn eindige representatie. Zowel het coderingsalgoritme als het decoderingsalgoritme zijn uitdrukbaar in PFOL. Het volstaat nu om de expressiviteit te bestuderen van enkel die PFOL-uitdrukbare vragen die eindige gegevensbanken op eindige gegevensbanken afbeelden. Om deze specifieke deelklasse van de PFOL-uitdrukbare vragen te onderzoeken, stellen we een tussentaal op, SPFOL genoemd, die precies deze vragen kan uitdrukken. We nemen vervolgens de taal SPFOL onder de loep en laten zien dat de expressiviteit van SPFOL

zeer nauw aanleunt bij de expressiviteit van de bevragingstaal SafeEuql, een taal die eerder is ontwikkeld om alle passer en liniaal constructies op eindige gegevensbanken te kunnen uitdrukken. Dit resultaat laat ons toe om te besluiten dat PFOL precies die vragen kan uitdrukken waarvoor de eindige representatie van de uitvoer "construeerbaar" is uit de eindige representatie van de invoer van die vraag. We merken op dat deze klasse van construeerbare vragen niet de volledige klasse van lineaire vragen uitdrukbaar in FO + poly dekt: we staven dit met een voorbeeld van een lineaire vraag die wel uitdrukbaar in FO + poly, doch niet construeerbaar is.

Nu de expressiviteit van PFOL bestudeerd is, komen we nog eens terug op de expressiviteit van uitbreidingen van FO + linear met lineaire operatoren. We vinden namelijk precies twee lineaire operatoren waarvan we aantonen dat FO + linear uitgebreid met deze lineaire operatoren precies dezelfde klasse van lineaire vragen kan uitdrukken als PFOL. Het is op dit moment nog een open probleem of er een uitbreiding van FO + linear bestaat met een eindig aantal operatoren die volledig is voor de lineaire vragen uitdrukbaar in FO + poly.

We proberen ook te laten zien dat de voorgestelde uitbreidingen van FO + linear niet alleen interessant zijn vanuit het oog van een theoreticus. Vrij recent is er een prototype van een geometrisch gegevensbanksysteem ontwikkeld dat gebaseerd is op het lineaire model met FO + linear als bevragingstaal. We bespreken hoe onze voorgestelde uitbreidingen van FO + linear, vertrekkende van een implementatie van FO + linear, kunnen worden geïmplementeerd.

We ronden uiteindelijk dit deel van de thesis af met enkele lineaire talen die volledig zijn voor de lineaire vragen uitdrukbaar in FO + poly.

De eenvoudigste manier om tot een volledige taal te komen, is het vinden van een algoritme dat kan beslissen of een FO + poly-uitdrukbare vraag lineair is of niet. We bewijzen echter dat een dergelijk algoritme niet bestaat.

We kunnen de eerder ontwikkelde lineariteitstest gebruiken om tot een volledige taal te komen. In weze komt die taal overeen met FO + poly, doch we gebruiken de lineariteitstest om te weten te komen of de uitvoer van een FO + poly formule op een semi-lineaire invoer semi-lineair is of niet. In het laatste geval geven we de lege verzameling als resultaat van die FO + poly formule op de invoer terug. Het is duidelijk dat deze manipulatie van FO + poly leidt tot een taal die compatibel en volledig is.

Een andere methode om tot een volledige taal te komen, is gebruik maken van PFOL formules die enerzijds de eindige representatie van een semi-lineaire verzameling berekenen, en anderzijds de semi-lineaire verzameling herberekenen uit zijn eindige representatie. Stel dat we een taal Q hebben die volledig is voor alle lineaire vragen die eindige gegevensbanken op eindige gegevensbanken afbeelden. De taal die dan bekomen wordt als de samenstelling van de decoderingsformule, ver-

volgens een formule van Q en tenslotte de coderingsformule, is duidelijk compatibel en volledig voor alle lineaire vragen uitdrukbaar in FO + poly. Voor de taal Q kunnen we de *polynomial-restricted queries* van Benedikt en Libkin nemen, waarvan zij hebben aangetoond dat deze vragen precies overeenkomen met de lineaire vragen die eindige gegevensbanken op eindige gegevensbanken afbeelden. Een andere mogelijkheid om tot een taal Q te komen, bestaat in het gebruiken van de FO + linear formule die beslist of een semi-lineaire verzameling eindig is. Zoals met de lineariteitstest, kunnen we FO + poly manipuleren zodat we een taal bekomen die precies alle lineaire vragen die eindige gegevensbanken op eindige gegevensbanken afbeelden kan uitdrukken.

We geven toe dat de volledige talen die we hierboven voorstellen niet bijster interessant zijn vanuit een praktisch oogpunt, maar ze vormen veeleer een rechtvaardiging voor het zoeken naar praktische bevragingtalen die alle lineaire vragen uitdrukbaar in FO + poly kunnen uitdrukken.

We beëindigen deze thesis met het aanstippen van enkele suggesties voor toekomstig onderzoek omtrent lineaire bevragingstalen.

Contents

1 Introduction **1**

2 Constraint Database Models **7**

2.1 The Euclidean Space: Terminology 7

2.2 The Polynomial Constraint Database Model 8

2.3 The Linear Constraint Database Model 16

3 Expressiveness of FO + linear **23**

3.1 FO + linear Expressible Properties of Semi-Linear Sets 24

3.2 Linear Queries on Unions of Affine Subspaces 36

3.3 Limitations of FO + linear 45

3.4 Expressibility in FO + linear is Undecidable 51

4 Decidability of Semi-Linearity of Semi-Algebraic Sets **55**

4.1 Property SL 56

4.2 Algorithmic Decompositions of Semi-Linear Sets 62

4.3 **A**-Semi-Linearity of Semi-Algebraic Sets is Decidable 68

4.4 **Z**-Semi-Linearity of Semi-Algebraic Sets is Decidable 77

5 Extensions of FO + linear **81**

5.1 FO + linear Extended with Operators 82

5.2 The Query Language PFOL 84

5.3 Expressiveness of PFOL 89

5.3.1 Finite Representations of Semi-Linear Sets 89

5.3.2 The Language SPFOL 94
5.3.3 Expressiveness of SPFOL 109
5.3.4 Expressiveness of PFOL 117
5.4 Expressiveness of FO + linear Extended with Operators 119
5.5 Query Languages Complete for FO + poly^{lin} 124

6 Discussion **127**
6.1 Main Results 127
6.2 Some Remarks on Implementing PFOL and FO + linear + $\mathcal{O}$ 128
6.2.1 On the Implementation of FO + linear 128
6.2.2 On the Implementation of FO + linear + $\mathcal{O}$ 130
6.2.3 On the Implementation of PFOL 131
6.3 Directions for Future Research 131

A Property SL and Semi-Linearity **135**

B Alternative Proof for Decidability of Semi-Linearity for Semi-Algebraic Sets **143**

Chapter 1

Introduction

Traditionally, database systems are used to store and manipulate alphanumerical data. In the early seventies, Codd [18] proposed the *relational database model* to deal with alphanumerical data in a structured way. Nowadays, most database systems are based on the relational database model as proposed by Codd. However, there is a growing need for database systems that can also deal with more general data than alphanumerical data, among which geometric data. *Geometric database systems* should be capable of representing and storing both geometric data and alphanumerical data. In addition, they should allow the manipulation of geometric data in their query language, and they should support geometric data in their implementation, for instance, with spatial access and indexing methods [24, 37, 41]. The need for geometric database systems is partially due to new database applications with a geometric nature [6, 51, 55, 58, 64], such as geographic information systems (GIS), geometric modeling systems (CAD), autonomous navigation, architectural and medical imaging, and simulation processing. The representation, storage, manipulation, and indexing of geometric data, however, cannot be supported adequately by database systems based on Codd's relational model (see, for instance, [23]). As a consequence, a new challenge for the database community was launched with as the purpose the development of a *geometric database model* that deals with both alphanumerical and geometric data in a sound and natural way. The research in this area is still ongoing, and much work has still to be done.

At this time, we can roughly categorize the existing geometric database models in two classes [60, 62]: the *geometric data type* (GDT) based models and the *constraint* based models. Both models have in common that they extend the relational model

to deal with geometric data.

The GDT-based approach extends Codd's relational model with a fixed set of geometric data types, and provides a set of geometric operators on these data types. Typically, these systems include data types for points, line segments, and convex polygons, and extend SQL with geometric operators on these data types to compute, for instance, intersections, the convex hull, the distance, and the topological boundary. A major drawback of these models is that they are application-dependent, since no set of geometric data types or operators that serves all application needs is yet known. For examples of GDT-based geometric database models, we refer to [1, 5, 13, 25, 27, 38, 39, 40, 43, 44, 68, 70] and references therein.

The constraint-based approach, introduced in the seminal paper of Kanellakis, Kuper, and Revesz [50], and adopted by [4, 49, 52, 61, 73], uses constraints as data in the relational model. They allow users to define geometric figures with constraints formulated as first-order logic formulae of a certain type. The geometric figure represented by such a formula consists of all points (in an appropriate Euclidean space) which satisfy the formula. The constraints most studied in this context are the polynomial constraints and the linear constraints. Since data of the relational model can be seen as constraints of the form $x = a$, where x is a variable ranging over the domain of alphanumerical values and a is an alphanumerical constant, one can view the constraint model as a generalization of the relational model. A natural query language to accompany these data models is the relational calculus extended with the class of constraints used to represent the geometric data. A major difficulty of this approach concerns implementation of these models and, only recently, efforts have been made to develop database systems based on the constraint model [12, 14, 31, 32].

The GDT-based models typically address geometric database applications in a fixed, lower dimensional space, such as geographic information systems and geometric modeling systems. Constraint models are not limited to a fixed dimension. They offer a declarative framework for the representation of arbitrary dimensional—possibly unbounded and topologically non-closed—geometric figures. As a consequence, queries within the constraint database model yield exact geometric results rather than approximated values. The generality of the geometric figures which are under consideration in constraint-based models is of relevance, although, at a first glance, practical applications are rarely situated in a dimension higher than 4. As an example, consider the following linear problem "Compute the position of a couple of shelves in a furnished room such that desks can still be sat at, and doors and drawers can still be opened." The solution to this problem (e.g., the coordinates of the mass-centers of the shelves) will define a geometric figure in six-dimensional space. Moreover that geometric figure will not be topologically closed, since open doors and drawers may not touch the shelves.

In the implementation of GDT-based models, the data structures for representing the different data types are selected in such a way that the various geometric operators can be computed as efficiently as possible using techniques from computational geometry [30, 36, 42, 66]. This approach garantuees good performance for evaluating queries, which will be hard to beat with constraint-based models. The geometric data used in constraint-based models are sometimes too general to ensure efficient implementation of the accompanying query language. For efficiency reasons, the linear constraint database model has received more attention than the polynomial constraint database model, lately. In both approaches, however, there is a clear need for geometric indexing and access techniques to make geometric database systems ready for general use.

Finally, constraint-based models have a formally defined semantics which makes them attractive from a theoretical point of view. These models allow the study of geometric databases and their properties in a less ad-hoc and more uniform way.

In this work, we concentrate on the *linear constraint database model*.

Various researchers have studied the expressive power of database query languages based on linear and polynomial constraints [2, 3, 4, 8, 10, 33, 34, 35, 53, 73, 74], but we are still far away from a precise insight in the nature of the queries expressible in these languages. Although the precise expressiveness of linear and polynomial constraint query languages has not yet been characterized satisfactorily, we have results on the non-expressibility of queries such as the parity query and the connectivity query in the polynomial constraint query language, hence also in the linear constraint query language, and the convex closure query and the distance query in the linear constraint query language. Furthermore, people started to investigate spatial aggregate functions (e.g., area and volume) and found out that these aggregate functions were undefinable in constraint query languages and that adding these functions to the query language created serious safety[1] problems [16, 67].

As a consequence of the non-expressibility of fundamental queries mentioned above, it is hard to claim that the linear constraint query language as initially proposed in the linear constraint data model can act as a general-purpose query language for the linear constraint data model. Clearly, there is a need for more powerful query languages which are still sound for the linear constraint data model. In practice, the design of these query languages seems to be difficult: extending the linear constraint query language results easily in a language with the same expressive power as the polynomial constraint query language which is, of course, not sound for the linear constraint database model [73].

The main contributions of this research dissertation can be summarized as follows:

[1] A constraint query language is called *sound* if a query of this language applied to a constraint database of a given type always results in a constraint database of the same type.

1. We provide a list of queries of which we show that they can be expressed in FO + linear, i.e., the relational calculus extended with linear constraints, and a tool which can be used to prove non-expressibility of a query in FO + linear. We show that the type of the coefficients of the linear constraints used in the linear constraint database model does influence certain results. We also prove that it is undecidable whether a linear query expressible in FO + poly, i.e., the relational calculus extended with polynomial constraints, can be expressed in FO + linear.

2. We propose a technique to decompose semi-linear sets (figures representable with linear constraints) into convex cells which is expressible in FO + poly. As a side-effect, we obtain algorithms, implementable in FO + poly, to compute a finite representation of an arbitrary semi-linear set and to recompute a semi-linear set from its finite representation.

3. We show that it is decidable whether a geometric figure definable with polynomial constraints is semi-linear.

4. We present two sound extensions of FO + linear for linear queries expressible in FO + poly. One extension can be seen as a bridge between GDT-based query languages and FO + linear, while the other extension results in a query language of which the expressive power can be characterized in terms of the ruler and compass constructions in the two-dimensional plane. We also show that there exist query languages which are complete for the linear queries expressible in FO + poly.

The outline of the remaining chapters is as follows.

In Chapter 2, we introduce the polynomial constraint database model and define the linear constraint database model as a restriction of the polynomial constraint database model. We argue that two natural restrictions of the polynomial constraint database model can be considered. We also establish the equivalence between finite unions of polyhedra and geometric figures representable with linear constraints, which further motivates the use of the linear constraint model.

In Chapter 3, we examine the expressiveness of the linear query language FO+linear. First, we present a list of general queries expressible in FO + linear, which act as "building blocks" for more sophisticated queries in the remainder of this work. In particular, we show that the notions of *dimension* and *regular point* of a semi-linear set can be expressed in FO + linear. We stress the importance of those two notions as they are the basis for the semi-linearity test developed in Chapter 4 and the finite representation technique of semi-linear sets developed in Chapter 5. Next, we positively illustrate the expressive power of FO + linear by establishing

that geometric notions such as orthogonality and parallelism can be expressed in FO+linear. We then argue that FO+linear is nevertheless *not* sufficiently expressive to be regarded as a general-purpose query language for the linear constraint database model. Concretely, we present a theorem that lifts the non-semi-linearity of certain semi-algebraic point-sets to the non-expressibility of closely related linear queries. We exhibit several examples of important linear queries which are proven to be inexpressible in FO + linear using the above theorem. Finally, we show that it is undecidable whether a linear query expressible in FO + poly can be expressed in FO + linear.

In Chapter 4, we show that "semi-linearity" is decidable for semi-algebraic sets. We observe that semi-linearity can be studied in the context of the liberal or the more restrictive limitation of the polynomial constraint database model to the linear constraint database model. We prove that semi-linearity of semi-algebraic sets in the more liberal sense can be decided by an FO+poly expression, while semi-linearity of semi-algebraic sets in the more restricted sense cannot be decided by an FO + poly expression. An interesting by-product of the proof techniques used is an algorithm to decompose an arbitrary semi-linear set into convex cells which is expressible in FO + poly.

In Chapter 5, we study extensions of FO + linear which remain sound for linear constraint databases and capture a broader class of linear queries expressible in FO + poly than FO + linear. We first propose a method to extend FO + linear with linear "operators." To illustrate the potential of this method, we exhibit two linear operators such that the extension of FO + linear with these operators has the same expressive power as the query language PFOL, another extension of FO + linear, which we study next. The query language PFOL is defined as FO+linear augmented with a limited amount of multiplicative power. We show that the PFOL-expressible queries which return finite outputs upon finite inputs are closely related to the queries expressible in SafeEuql [63], a query language which was designed to capture the ruler and compass constructions in the two-dimensional plane. Thereto, we study a finite representation of arbitrary semi-linear sets which can be computed within PFOL. We conclude this chapter with providing query languages which are complete for the linear queries expressible in FO + poly.

In Chapter 6, we briefly describe approaches to implement the extensions of FO + linear studied in Chapter 5. This should motivate the use of these languages in implementations of the linear constraint model. We also propose some directions for future research.

Chapter 2

Constraint Database Models

In this chapter, we provide the necessary background on the polynomial and linear constraint database models. We explain how geometric data can be described in a finite way using polynomial and linear constraints. The geometric figures that can be described with linear constraints are precisely those figures that can be described as a finite union of open polyhedra. We introduce 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 database model. We argue that two natural restrictions of the polynomial constraint database model to a linear constraint database model are possible. We study the notion of query in the context of constraint database models and define for each model a natural calculus-like query language and an equivalent algebra-like query language. Since the linear constraint database model is a sub-model of the polynomial constraint database model, we start with the latter. First, though, since we work in the n-dimensional Euclidean space $\mathbf{R}^n$ for the remainder of this dissertation, we briefly introduce some terminology about $\mathbf{R}^n$.

2.1 The Euclidean Space: Terminology

From now on, we work in the n-dimensional Euclidean space $\mathbf{R}^n$. The *canonical coordinate basis* in $\mathbf{R}^n$ has origin $\vec{0}$ and unit vectors $\vec{e}_1, \ldots, \vec{e}_n$ where the vector $\vec{e}_i$, $1 \leq i \leq n$, has ith coordinate 1 and all other coordinates 0.

A *linear subspace* of $\mathbf{R}^n$ is a subset of $\mathbf{R}^n$ that is closed under vector addition and scalings. The *dimension* of a linear subspace is the smallest number k for which there exists k vectors which span the linear subspace. For example, the linear subspaces of $\mathbf{R}^3$ are the origin $\vec{0}$ of the canonical coordinate system (which has dimension 0), all lines of $\mathbf{R}^3$ through $\vec{0}$ (which have dimension 1), all planes of $\mathbf{R}^3$ through $\vec{0}$ (which have dimension 2), and $\mathbf{R}^3$ itself (which has of course dimension 3). A *d-dimensional coordinate subspace* of $\mathbf{R}^n$, $0 \leq d \leq n$, is any linear subspace of $\mathbf{R}^n$ generated by exactly d of the canonical basis vectors $\vec{e}_1, \ldots, \vec{e}_n$.

An *affine subspace* of $\mathbf{R}^n$ is a translation of a linear subspace of $\mathbf{R}^n$. Hence, the affine subspaces of $\mathbf{R}^3$ are all points of $\mathbf{R}^3$, all lines of $\mathbf{R}^3$, all planes of $\mathbf{R}^3$, and $\mathbf{R}^3$ itself. A *hyperplane* of $\mathbf{R}^n$ is an $(n-1)$-dimensional affine subspace of $\mathbf{R}^n$. A *coordinate hyperplane* of $\mathbf{R}^n$ is an $(n-1)$-dimensional coordinate subspace of $\mathbf{R}^n$.

Two affine subspaces of $\mathbf{R}^n$ are *parallel* if either one is contained in the other or they have no points in common. Two affine subspaces of $\mathbf{R}^n$ are called *orthogonal* if their corresponding linear subspaces (obtained by translating the affine subspaces to the origin of the coordinate system) are orthogonal. Two linear subspaces of $\mathbf{R}^n$ are orthogonal if each vector $\vec{x}$ of the first subspace is orthogonal to each vector $\vec{y}$ of the second subspace, i.e., if $\vec{x}.\vec{y} = 0$ [1]. If S is a linear subspace of $\mathbf{R}^n$ of dimension k, then the set $S^\perp$ of *all* vectors of $\mathbf{R}^n$ orthogonal to all vectors of S is a linear subspace of $\mathbf{R}^n$ of dimension $n-k$.

Let $S \subseteq \mathbf{R}^n$. The *affine support* of S is the lowest dimensional affine subspace of $\mathbf{R}^n$ containing S.

2.2 The Polynomial Constraint Database Model

In the polynomial constraint database model [50], we consider as geometric data all geometric figures definable in elementary geometry [11], i.e., $\langle \mathbf{R}, \leq, \times, +, 0, 1 \rangle$, the first-order logic over the real numbers with addition and multiplication. The rationale behind this approach is that the first-order theory of the reals is decidable by means of a strong form of effective quantifier elimination [6, 19], and that, consequently, many properties of figures definable in elementary geometry are decidable, too [46, 69, 71].

We now explain how geometric data can be represented with first-order logic formulae over the reals. Assume a totally ordered infinite set of variables called *real variables*, which range over the real numbers. Define a *polynomial constraint*

[1] Let $\vec{x}$ and $\vec{y}$ be two vectors in $\mathbf{R}^n$. Then, the dot-product $\vec{x}.\vec{y}$ is defined as $x_1y_1 + \ldots + x_ny_n$.

term as a polynomial in real variables with integer coefficients[2]. Then, an *atomic polynomial constraint formula* is built from a polynomial constraint term using binary comparison relations, i.e., $t\,\theta\,0$ with t a polynomial constraint term and $\theta \in \{=, <, >, \leq, \geq, \neq\}$. A *polynomial constraint formula*[3] is defined as a well-formed first-order logic formula built from atomic polynomial constraint formulae, i.e.,

- 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 $x_1, \ldots, x_n$ defines a point-set

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

in n-dimensional Euclidean space $\mathbf{R}^n$ in the standard manner. Point-sets defined by polynomial constraint formulae are called *semi-algebraic sets*.

Example 2.1 The polynomial constraint formula

$$(\exists u)(0 \leq x \wedge x \leq 10 \wedge 25u = x^2 - 10x + 50 \wedge y^2 + z^2 = u^2)$$

describes the geometric figure shown in Figure 2.1. ■

Notation 2.2 For convenience, we shall frequently use vector notation in polynomial constraint formulae. Atoms involving vector notation must be interpreted coordinate-wise. For instance, in three-dimensional space the expression $\vec{x} = \alpha\vec{y} + \beta\vec{z}$ should be interpreted as the polynomial constraint formula $x_1 = \alpha y_1 + \beta z_1 \wedge x_2 = \alpha y_2 + \beta z_2 \wedge x_3 = \alpha y_3 + \beta z_3$. In particular, $\neg(\vec{x} = \vec{0})$ indicates that $\vec{x}$ is not the origin of the coordinate system, whereas $\vec{x} \neq \vec{0}$ denotes that *none* of the coordinates of $\vec{x}$ equals 0. As usual, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$, and $(\forall x)\varphi$ will be used as abbreviations for $\neg(\neg\varphi \wedge \neg\psi)$, $\neg\varphi \vee \psi$, $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, and $\neg(\exists x)\neg\varphi$, respectively. Furthermore, we shall also use polynomials with rational and real algebraic coefficients in polynomial constraint formulae, since these polynomial constraint formulae can be rewritten as polynomial constraint formulae that only use polynomials with integer coefficients. ■

[2]In order to obtain formulae that are finitely representable, only coefficients that are finitely representable (e.g., integers) may be allowed.

[3]Observe that "polynomial constraint terms" and "polynomial constraint formulae" correspond to "real terms" and "real formulae", respectively, which are often used in the context of the real closed field.

Figure 2.1: Example of a semi-algebraic figure.

By definition, semi-algebraic sets are finitely representable by means of polynomial constraint formulae. It must be noted that several polynomial constraint formulae can represent the same semi-algebraic set, as illustrated by the following example.

Example 2.3 The two polynomial constraint formulae define the same area in the plane:

- $(\exists x_3)(\exists x_4)(x_3^2 + x_4^2 = 100 \wedge (x_3 - x_1)^2 + (x_4 - x_2)^2 < 1)$; and
- $x_1^2 + x_2^2 > 81 \wedge x_1^2 + x_2^2 < 121$. ■

In fact, every semi-algebraic set can be represented by an infinite number of polynomial constraint formulae. On the contrary, every polynomial constraint formula unambiguously defines one semi-algebraic set.

By the quantifier elimination theorem of Tarski [71], it is always possible to represent a semi-algebraic set by a quantifier-free polynomial constraint formula. (Notice that the second formula in Example 2.3 is quantifier-free.) The same theorem also guarantees decidability of the equivalence of two polynomial constraint formulae.

Notice that the intersection and the union of two semi-algebraic sets of $\mathbf{R}^n$ can be interpreted as, respectively, the conjunction and disjunction of the polynomial constraint formulae representing those semi-algebraic sets; the complement of a semi-algebraic set as the negation of the polynomial formula representing that semi-algebraic set; and, finally, the geometric projection of a semi-algebraic set on a coordinate subspace as the existential quantification of the appropriate free variables in the polynomial constraint formula representing that semi-algebraic set. As a consequence, semi-algebraic sets are closed under intersection, union, complement, and projection.

We next introduce the polynomial constraint database model. In essence, the polynomial constraint database model is an extension of the relational data model where a relation, besides columns that store values of some alphanumerical data type, can have one extra geometric column of type semi-algebraic set. In contrast with the alphanumerical data columns, there is a sharp distinction between what is stored in a geometric column (finitely representable polynomial constraint formulae) and the meaning of the stored data (possibly infinite point-sets, which may even be unbounded). In the next two paragraphs, we give the formal definitions.

A *polynomial constraint database scheme*, $\mathcal{S}$, is a finite set of *relation names*. We associate with each relation name, R, a type which is a pair of natural numbers, $[m, n]$, where m denotes the number of alphanumerical columns and n the dimension of the single geometric column of R. A polynomial constraint database scheme has type $[m_1, n_1; \ldots ; m_k, n_k]$ if the scheme consists of relation names, $R_1, \ldots , R_k$, respectively of type $[m_1, n_1], \ldots , [m_k, n_k]$. A *syntactic polynomial constraint relation* of type $[m, n]$ is a finite set of tuples of the form

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

with $v_1, \ldots , v_m$ alphanumerical values of some domain[4], $\mathbf{U}$, and $\varphi(x_1, \ldots , x_n)$ a polynomial constraint formula with n free real variables. As argued before, we may assume without loss of generality that this formula is quantifier-free. A *syntactic polynomial constraint database instance* is a mapping, $\mathcal{I}$, assigning to each relation name, R, of a scheme, $\mathcal{S}$, 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

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

This subset of $\mathbf{U}^m \times \mathbf{R}^n$ can be interpreted as a possibly infinite $(m + n)$-ary relation, called *semantic polynomial constraint relation*, the tuples of which are called semantic polynomial constraint tuples. The semantics of a syntactic polynomial constraint database instance, $\mathcal{I}$, over a polynomial constraint database scheme, $\mathcal{S}$, is the mapping, I, assigning to each relation name, R, in $\mathcal{S}$ the semantic polynomial constraint relation $I(\mathcal{I}(R))$.

Example 2.4 The example in Figure 2.2 shows a polynomial constraint database containing geographical information about Belgium, represented with linear constraints. ∎

[4]Typically, in relational database systems, this domain consists of integers, characters, dates, etc.

Regions

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

Cities

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

Rivers

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

Figure 2.2: Example of a polynomial constraint database database.

In traditional database theory, a query is usually defined as a computable mapping from databases to databases. In geometric models, such as the polynomial constraint database model, the picture is somewhat more complicated, since queries can be viewed both at the syntactic level and the semantic level. Therefore, given an input scheme $\mathcal{S}_{\text{in}}$ and an output scheme $\mathcal{S}_{\text{out}}$, a query is a mapping of the polynomial constraint database instances of $\mathcal{S}_{\text{in}}$ to the polynomial constraint database instances of $\mathcal{S}_{\text{out}}$, both at the syntactic and the semantic level. Moreover, at the syntactic level, a query must be computable[5].

We associate with every query a type

$$[m_1, n_1; \ldots ; m_k, n_k] \to [m, n]$$

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

An obvious query language accompanying the polynomial constraint database model is obtained by adding to the language of the polynomial constraint formulae the following:

1. a totally ordered infinite set of variables, called *value variables*, disjoint from the set of real variables, which range over the domain $\mathbf{U}$ of alphanumerical data values;
2. atomic formulae of the form $v_1 = v_2$, with v_1 and v_2 value variables;
3. atomic formulae of the form $R(v_1, \ldots , v_m; x_1, \ldots , x_n)$, with R a relation name of type $[m, n]$, $v_1, \ldots , v_m$ value variables, and $x_1, \ldots , x_n$ real variables; and
4. existential quantification of value variables.

In the literature, this query language is commonly known as the *polynomial constraint calculus*, or, for short, FO + poly. A query of type $[m_1, n_1; \ldots ; m_k, n_k] \to [m, n]$ 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 $[m_1, n_1; \ldots ; m_k, n_k]$, $\{(v_1, \ldots , v_m; x_1, \ldots , x_n) \mid \varphi(v_1, \ldots , v_m; x_1, \ldots , x_n)\}$ equals the corresponding output database, which is of type $[m, n]$. As is shown by [9, 34, 53], not every query is expressible in FO + poly.

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

[5]Although it is not within the scope of this dissertation, a query should satisfy, at the semantic level, some *genericity* condition, as first studied in [15] for queries on classical relational databases, and generalized to queries on geometric databases in [61].

Example 2.5 Consider the polynomial constraint database of Example 2.4. A query on this database is *"Give the name and position of all cities that lie within unit distance from the border between Flanders and the Wallonia."* This query of type $[1,2;1,2;1,2] \to [1,2]$ can be expressed with the formula

$$(\exists u)(\exists v)\text{Cities}(c,x,y) \wedge \text{Regions}(\text{'Flanders'};u,v) \wedge \\ \text{Regions}(\text{'Wallonia'};u,v) \wedge (x-u)^2+(y-v)^2 < 1$$

in FO + poly. ■

Due to the existence of quantifier elimination algorithms for polynomial constraint formulae, every FO + poly-expressible query is effectively computable, and yields a polynomial constraint database as result [50].

We conclude this section by presenting an equivalent *polynomial constraint algebra* for FO + poly, the polynomial constraint calculus [61]. This polynomial constraint algebra maps polynomial constraint database instances to polynomial constraint database instances *at the syntactic level*. Thereto, we translate the polynomial constraint calculus formulae into equivalent polynomial constraint algebra expressions which have an operational semantics on the syntactic level.

The operators of the polynomial constraint algebra are as follows. We use the following terminology: given a syntactic tuple

$$t = (v_1,\ldots,v_m;\varphi(x_1,\ldots,x_n)),$$

we denote the *alphanumerical part* of t, $(v_1,\ldots,v_m)$, by $\text{val}(t)$ and the *geometric part* of t, $\varphi(x_1,\ldots,x_n)$, by $\text{geom}(t)$.

- Let r_1 and r_2 be syntactic relations of type $[m,n]$ and assume, without loss of generality, that the geometric parts of all syntactic tuples in r_1 and r_2 use the same free variables.
  - The *union*, denoted $r_1 \cup r_2$, is the standard set-theoretic union.
  - The *difference*, denoted $r_1 - r_2$, is the syntactic relation

    $$\{(\text{val}(t);\varphi_t(x_1,\ldots,x_n) \mid t \in r_1\},$$

    where $\varphi_t(x_1,\ldots,x_n)$ stands for

    $$\bigvee_{\substack{t_1 \in r_1 \\ \text{val}(t_1)=\text{val}(t)}} \text{geom}(t_1) \wedge \neg \bigvee_{\substack{t_2 \in r_2 \\ \text{val}(t_2)=\text{val}(t)}} \text{geom}(t_2).$$

  - The *intersection*, denoted $r_1 \cap r_2$ is defined in the standard way as $r_1 - (r_1 - r_2)$.

- Let r_1 and r_2 be syntactic relations of types $[m_1, n_1]$ and $[m_2, n_2]$, respectively, and assume, without loss of generality, that no free real variable occurs in the geometric part of a syntactic tuple in both r_1 and r_2.

  The *Cartesian product*, denoted $r_1 \times r_2$, is the syntactic relation

  $$\{(\mathrm{val}(t_1), \mathrm{val}(t_2); \mathrm{geom}(t_1) \wedge \mathrm{geom}(t_2)) \mid t_1 \in r_1,\ t_2 \in r_2\},$$

  which is of type $[m_1 + m_2, n_1 + n_2]$.

- Let r be a syntactic relation of type $[m, n]$, and assume, without loss of generality, that the geometric parts of all syntactic tuples in r use the same free real variables $x_1, \ldots, x_n$.

  - For $1 \leq i, j \leq m$, the *value selection*, denoted $\sigma_{i=j}(r)$, is the syntactic relation

    $$\{t \in r \mid \mathrm{val}(t)(i) = \mathrm{val}(t)(j)\}.$$

    For $a \in \mathbf{U}$, the *constant value selection*, denoted $\sigma_{i=a}(r)$, is the syntactic relation

    $$\{t \in r \mid \mathrm{val}(t)(i) = a\}.$$

  - For φ a quantifier-free polynomial constraint formula with free variables $x_1, \ldots, x_n$, the *geometric selection*, denoted $\sigma_\varphi(r)$, is the syntactic relation

    $$\{(\mathrm{val}(t); \mathrm{geom}(t) \wedge \varphi) \mid t \in r\}.$$

  - For $1 \leq i_1, \ldots, i_p \leq m$, the *value projection*, denoted $\pi_{i_1,\ldots,i_p}(r)$, is the syntactic relation

    $$\{(\mathrm{val}(t)(i_1), \ldots, \mathrm{val}(t)(i_p); \mathrm{geom}(t)) \mid t \in r\},$$

    which is of type $[p, n]$.

  - For $1 \leq i_1, \ldots, i_p \leq n$, the *geometric projection*, denoted $\pi_{x_{i_1},\ldots,x_{i_p}}(r)$, is the syntactic relation

    $$\{(\mathrm{val}(t); \pi_{x_{i_1},\ldots,x_{i_p}}(\mathrm{geom}(t))) \mid t \in r\},$$

    which is of type $[m, p]$, where, for a polynomial constraint formula φ with free variables $x_1, \ldots x_n$, $\pi_{x_{i_1},\ldots,x_{i_p}}(\varphi)$ is defined as (the quantifier-free equivalent of) the polynomial constraint formula

    $$(\exists x_1) \ldots (\exists x_n)(\varphi(x_1, \ldots, x_n) \wedge \bigwedge_{k=1}^{p} y_k = x_{i_k}),$$

    with $y_1, \ldots, y_p$ real variables satisfying $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_p\} = \emptyset$.

- For each k, the constant relations $\mathbf{U}^k$ and $\mathbf{R}^k$ denote the "maximal" syntactic relation of type $[k, 0]$ and $[0, k]$, respectively.

Polynomial constraint algebra expressions are obtained by applying the above operators to relation names.

Example 2.6 The query in Example 2.5 on the polynomial constraint database of Example 2.4 can be expressed in the polynomial constraint algebra as

$$\pi_1 \, \pi_{x_1,x_2} \, \sigma_\varphi \, \sigma_{2=\text{`Wallonia'}} \, \sigma_{3=\text{`Flanders'}} \, (\text{Cities} \times \text{Regions} \times \text{Regions})$$

where φ equals $x_3 = x_5 \wedge x_4 = x_6 \wedge (x_1 - x_3)^2 + (x_2 - x_4)^2 < 1$. ■

Using standard techniques, we can establish the following result:

Proposition 2.7 *Every* FO + poly *expression can be converted effectively into a polynomial constraint algebra expression and vice-versa, in such a way that both express the same mapping from polynomial constraint database instances to polynomial constraint database instances, respectively at the semantic and syntactic level.*

As queries expressed by polynomial constraint algebra expressions are obviously computable at the syntactic level, Proposition 2.7 also establishes that FO+poly and the polynomial constraint algebra are indeed geometric query languages as discussed earlier in this section.

2.3 The Linear Constraint Database Model

From the polynomial constraint database model, the linear constraint database model can be obtained by only considering polynomial constraint formulae with linear polynomials. There are two obvious ways to obtain this restriction.

In the most restrictive approach, the degree of the polynomials involved is restricted to 1, as a consequence of which all the linear polynomials considered have integer coefficients. So, we consider only atomic polynomial constraint formulae that are built from polynomial constraint terms that are linear polynomials; these atomic polynomial constraint formulae are called *atomic* **Z**-*linear constraint formulae*, and the polynomial constraint terms from which they are built are called **Z**-*linear constraint terms*. Atomic **Z**-linear constraint formulae are of the form $\sum_{i=1}^{n} a_i x_i \, \theta \, a$, where $x_1, \ldots, x_n$ are real variables, $a_1, \ldots, a_n$ are integer coefficients, a is an integer, and θ is one of $=, >, \geq, <, \leq$, and $\neq$. Polynomial constraint formulae build of atomic **Z**-linear constraint formulae are called **Z**-*linear constraint formulae*. Clearly, **Z**-linear constraint formulae are equivalent to first-order formulae over the real numbers with addition only.

Remark 2.8 We shall frequently use **Z**-linear constraint formulae in which the linear polynomials have rational coefficients, since these formulae can be easily rewritten into formulae in which the linear polynomials only have integer coefficients. ■

A **Z**-linear constraint formula $\varphi(x_1, \ldots, x_n)$ with free real variables $x_1, \ldots, x_n$ defines a geometric figure $\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$ in n-dimensional Euclidean space $\mathbf{R}^n$ by letting real variables range over the real numbers. Semi-algebraic sets defined in this way are called **Z**-*semi-linear sets*.

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

Example 2.9 The polynomial constraint database considered in Example 2.4 and shown in Figure 2.2 contains only **Z**-linear constraint formulae to describe the geometric information, whence it is a **Z**-linear constraint database. ■

Queries in the context of **Z**-linear constraint databases are called **Z**-*linear queries* and the corresponding restriction of the query language FO + poly is denoted FO + **Z**-linear. Using algebraic computational techniques for the elimination of variables in sets of linear equations and inequalities [47, 48, 54], every FO + **Z**-linear query yields a **Z**-linear constraint database as a result by substituting the definition of the input relation, which are **Z**-linear constraint formulae, into the FO + **Z**-linear query.

Example 2.10 A (simple) example of a **Z**-linear query on the linear constraint database of Example 2.9 is *"Find all cities that lie on a river and give their name and position."* This query of type $[1, 2; 1, 2; 1, 2] \to [1, 2]$ which can be expressed as

$$(\exists r)\text{Cities}(c, x, y) \wedge \text{Rivers}(r, x, y)$$

in FO + **Z**-linear. ■

A more liberal restriction of the polynomial constraint database model to a linear constraint database model is obtained when the linear polynomials considered may have arbitrary *real algebraic coefficients*. In this case, we speak about **A**-linear constraint terms, atomic **A**-linear constraint formulae, **A**-linear constraint formulae, **A**-semi-linear sets, **A**-linear constraint relations, **A**-linear constraint databases, **A**-linear queries, and the query language FO + **A**-linear.

Example 2.11 Let a and b be two positive real algebraic numbers and consider the polynomial constraint formula

$$(x+a)^2+y^2=b^2 \wedge (x-a)^2+y^2=b^2 \wedge z \geq 0 \wedge z \leq 4.$$

The above formula, though non-linear, represents an **A**-semi-linear set in three-dimensional Euclidean space $\mathbf{R}^3$ for all possible real algebraic numbers a and b.

Indeed, the above formula describes the intersection of two parallel cylinders with radius b and height 4, one with central axis $x=-a \wedge y=0$ and the other with central axis $x=a \wedge y=0$. The intersection is either empty (if $a>b$) or the line segment $x=0 \wedge y=0 \wedge z \geq 0 \wedge z \leq 4$ on the z-axis (if $a=b$) or the set described by

$$(x=0 \wedge y=-\sqrt{b^2-a^2} \wedge z \geq 0 \wedge z \leq 4) \vee \\ (x=0 \wedge y=\sqrt{b^2-a^2} \wedge z \geq 0 \wedge z \leq 4),$$

consisting of two line segments parallel to the z-axis (if $a<b$.)

For example, if $a=4$ and $b=5$, then S is the set described by

$$(x=0 \wedge y=-3 \wedge z \geq 0 \wedge z \leq 4) \vee (x=0 \wedge y=3 \wedge z \geq 0 \wedge z \leq 4),$$

which is **Z**-semi-linear. (The projection of this set on the xy-plane is shown in Figure 2.3).

Figure 2.3: Projection of the **Z**-semi-linear set of Example 2.11 on the xy-plane.

If $a=1$ and $b=2$, then S is the set described by

$$(x=0 \wedge y=-\sqrt{3} \wedge z \geq 0 \wedge z \leq 4) \vee (x=0 \wedge y=\sqrt{3} \wedge z \geq 0 \wedge z \leq 4),$$

which is **A**-semi-linear, but not **Z**-semi-linear. ■

For most results presented here, there will be no need to make the distinction between linear polynomials with integer or real algebraic coefficients. Therefore, we shall speak in general about linear constraint terms, atomic linear constraint formulae, linear constraint formulae, semi-linear sets, linear constraint relation, linear constraint database, linear queries, and FO+linear whenever the distinction between integer or real algebraic coefficients has no influence on the proposed results. We stress, however, that the distinction between integer and real algebraic coefficients is not just a technical one: Chapter 4 shows that there exists interesting properties which are decidable by an FO + poly expression in the **A**-linear case, but not in the **Z**-linear case.

As in the case of semi-algebraic sets, the intersection and the union of two semi-linear sets of $\mathbf{R}^n$ can be interpreted as, respectively, the conjunction and disjunction of the linear constraint formulae representing those semi-linear sets; the complement of a semi-linear set denotes the negation of the linear formula representing that set; and, finally, the geometric projection of a semi-linear set on a coordinate subspace can be regarded as the existential quantification of the appropriate free variables in the linear constraint formula representing that semi-linear set. As a consequence, semi-linear sets are closed under intersection, union, complement, and projection.

Next, the above closure properties allow us to establish an alternative characterization of semi-linear sets.

A *polyhedron* in a Euclidean space (of arbitrary dimension) is defined as a finite intersection of closed half-spaces. Obviously, a polyhedron that can be defined in terms of half-spaces of which the bounding hyperplanes can be described by equations with integer (real algebraic) coefficients is **Z**-semi-linear (**A**-semi-linear). An *open polyhedron* is the topological interior of a polyhedron with respect to the lowest dimensional affine subspace containing that polyhedron. For instance, a point, an open half-line (which is topologically open within its supporting line), an open triangle (which is topologically open within its supporting two-dimensional plane), and an open cube with one face at infinity are open polehydra in three-dimensional space. Open polyhedra can always be described as the intersection of an appropriate set of open and closed half-spaces. The closed half-spaces are only needed to define the affine support of the polyhedron. Thus, when an open polyhedron has maximal dimension, open half-spaces suffice to describe it. In case a polyhedron is bounded, we speak of a *polytope* rather than a polyhedron. As an example, points, line segments, and convex[6] polygons are the only polytopes in two-dimensional space. The following result is easy to prove:

[6]Let S be a subset of $\mathbf{R}^n$. Then S is *convex* when, for every two points of S, the line segment defined by these points is fully contained in S.

Proposition 2.12 *Semi-linear sets and finite unions of open semi-linear polyhedra represent the same class of geometric figures. Bounded semi-linear sets and finite unions of open semi-linear polytopes are equivalent.*

Proof. Since open semi-linear polyhedra can be defined in terms of open and closed half-spaces that can clearly be described by atomic linear constraint formulae, and since the geometric operations intersection and union can be interpreted as the logical connectives $\wedge$ and $\vee$, respectively, it follows clearly that finite unions of open semi-linear polyhedra are semi-linear sets.

Since polytopes are bounded polyhedra, it is obvious that finite unions of open semi-linear polytopes are semi-linear bounded sets.

Conversely, an atomic linear constraint formula represents either an open or closed half-space, or a hyperplane (which is the intersection of two closed half-spaces), or, finally, the complement of a hyperplane (which is the union of two open half-spaces). So, clearly, an atomic linear constraint formula can be represented by one or two open semi-linear polyhedra. Since the geometric operations intersection and union can be interpreted as the logical connectives $\wedge$ and $\vee$, respectively, it follows, from a simple induction argument on the structure of a linear constraint formula, that any semi-linear set defined by a linear constraint formula *without* quantifiers and negation can alternatively be defined as a finite union of open semi-linear polyhedra. This result extends to semi-linear sets defined by *general* linear constraint formulae, as the quantifiers can be eliminated [47, 48, 54], and linear constraint formulae with negation can easily be rewritten as linear constraint formulae without negation.

A bounded semi-linear set can be written as a finite union of open semi-linear polyhedra. Because of the boundedness of the semi-linear set, each polyhedron is bounded, and, therefore, a polytope. ■

The above characterization allows us to conclude that most geometric data types found in the literature are sub-types of the semi-linear sets. Güting [39, 40] in his geo-relational algebra proposes the geometric data types *point*, *line*, and *polygon*, which can be seen as 0-, 1-, and two-dimensional polytopes, respectively.[7] Egenhofer [21] in his geometric data representation model proposes as basic objects *simplices*, which are special kinds of polytopes.

For FO + poly, the polynomial constraint calculus, we showed that there exists an equivalent polynomial constraint algebra. This polynomial constraint algebra can be restricted in the obvious way to obtain algebras which express **Z**-linear queries and **A**-linear queries, and such that we can state the following result:

[7] The polygons considered by Güting are not necessarily convex, but can always be decomposed into convex polygons.

Proposition 2.13 *Every* FO + linear *expression can be converted effectively into a linear constraint algebra expression and vice-versa, in such a way that both express the same mapping from linear constraint database instances to linear constraint database instances, respectively at the semantic and syntactic level.*

We point out that Afrati et al. [4] have shown that FO + linear is *not* complete for the linear queries definable within FO + poly. More concretely, Afrati et al. proved the following result:

Proposition 2.14 [4] *The Boolean query on semi-linear sets S of* $\mathbf{R}$ *which decides whether there exist u and v in S with $u^2 + v^2 = 1$, is not definable in* FO + linear.

Even though the query in Proposition 2.14 involves a non-linear computation in order to evaluate it, it is a linear query because it is a Boolean query, and therefore Proposition 2.14 suffices to establish the incompleteness of FO + linear for the linear queries definable in FO + poly. We shall denote the class of $\mathbf{Z}$-linear queries definable in FO + poly by FO + poly$^{\mathbf{z}\text{-}lin}$ and the class of $\mathbf{A}$-linear queries definable in FO + poly by FO + poly$^{\mathbf{a}\text{-}lin}$. When the distinction between the class of FO + poly$^{\mathbf{z}\text{-}lin}$ and FO + poly$^{\mathbf{a}\text{-}lin}$ queries is not important, we will speak of the class of FO + poly^{lin} queries. Various questions now arise, among which the following.

- Is it decidable whether an FO + poly query is an FO + poly$^{\mathbf{a}\text{-}lin}$ or an FO + poly$^{\mathbf{z}\text{-}lin}$ query?
- Is FO + linear an adequate linear query language for the linear constraint database model? Which FO + poly$^{\mathbf{a}\text{-}lin}$ (FO + poly$^{\mathbf{z}\text{-}lin}$) queries are expressible in FO + linear?
- Can we find linear query languages that are complete for the FO + poly$^{\mathbf{a}\text{-}lin}$ (FO + poly$^{\mathbf{z}\text{-}lin}$) queries?
- Are there interesting sub-classes of the FO + poly^{lin} queries for which there exists a natural query language with a nice geometric interpretation?

We try to answer these questions in the following chapters.

We conclude this section with the following remark. Although alphanumerical data are indispensable in most practical geometric database applications, it is of no use both in the study of geometric properties of semi-linear and semi-algebraic sets and in the examination of the expressiveness of the query languages FO + poly and FO + linear. Therefore, in the remainder of this dissertation, we shall focus on *purely geometric* constraint databases, i.e., constraint databases in which the constraint

relations have type $[0, n]$. As a consequence, we shall omit the alphanumerical part in the query languages FO + poly and FO + linear (and extensions thereof), and concentrate on the expressibility of *purely geometric queries*, i.e., queries from purely geometric constraint databases to purely geometric constraint databases.

Chapter 3

On the Expressiveness of FO + linear

In this chapter, we focus on the expressiveness and limitations of the linear query language FO + linear.

The FO + linear formulae we present in the first part of this chapter act as "building blocks" to compose FO + linear formulae for more sophisticated linear queries in the remainder of this work. This is illustrated in the second part of this chapter, where we show that the geometric properties *parallelism* and *orthogonality*, in the context of semi-linear sets consisting of affine subspaces, are expressible in FO + linear. Moreover, we study the notions of *dimension* and *regular point* of a semi-linear set, and show they can be expressed in FO + linear. This result is of importance, as the dimension and regular points of a semi-linear set form the basis for a decomposition technique of semi-linear sets that is used to obtain (i) an algorithm, which will be developed in Chapter 4, that decides whether a semi-algebraic set is semi-linear, and (ii) a method, which will studied in Chapter 5, to "lift" query languages defined on finite databases to query languages defined on linear constraint databases, which, at the end, will result in a query language that is complete for the class of FO + poly^{lin} queries.

In the second part of this chapter, we argue that FO + linear is nevertheless *not* sufficiently expressive to be regarded as a general-purpose linear query language for the linear constraint database model. To justify this claim, we exhibit a theorem stating that the non-semi-linearity of certain semi-algebraic sets can be "lifted" to

the non-expressibility of closely related linear queries. Armed with this tool, we show that certain linear queries, indispensable in most geometric database applications, can*not* be computed in FO + linear.

We conclude this chapter with a result stating that it is undecidable whether an FO + poly^{lin} query can be expressed in FO + linear. We also give examples of subclasses of the FO + poly^{lin} queries for which it remains undecidable whether queries of these subclasses can be expressed in FO + linear. This result shows that, to study the expressibility of certain queries in FO + linear, the "query-by-query" approach we took in the first two sections can make sense.

3.1 FO + linear Expressible Properties of Semi-Linear Sets

Since the query language FO + linear plays a key role in the upcoming chapters, we try to make the reader more comfortable with this query language and its expressive power by showing for some fundamental queries of topological or geometric nature how they can be expressed in FO + linear. In particular, we show that the *dimension* of a semi-linear set can be computed in FO + linear. We also introduce the notion of *regular point* of an arbitrary semi-algebraic set. We then show that, for semi-linear sets, the set of regular points can be computed in FO + linear. The notions dimension and regular point are the foundations of a decomposition technique of semi-linear sets which reveals to some extent the geometric structure of a semi-linear set.

To simplify our discussions, we assume that, in all the queries below, the input database consists of one relation name S of an arbitrary purely geometric type $[0, n]$.

We start with observing that semi-linear sets are closed under *translations*. Clearly, the FO + linear expression

$$(\exists \vec{u}) S(\vec{u}) \wedge \vec{z} = \vec{y} - \vec{x} + \vec{u}$$

computes the translation defined by $\overrightarrow{xy}$ [1] of the semi-linear set S.

The property "being symmetric with respect to a point" can be described in terms of translations. We say that a semi-linear set S of $\mathbf{R}^n$ is symmetric with respect to the point $\vec{p}$ of $\mathbf{R}^n$ if, for each point $\vec{x}$ of S, the point $\vec{p} + \overrightarrow{xp}$ belongs to S. Clearly, this property can be expressed in FO + linear by the sentence

$$(\forall \vec{x})(S(\vec{x}) \Rightarrow (\exists \vec{y})(\vec{y} = 2\vec{p} - \vec{x} \wedge S(\vec{y}))).$$

[1] The vector $\overrightarrow{xy}$ is defined as $\vec{y} - \vec{x}$.

The FO + linear expression

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

decides whether S consists of *isolated points* only. For semi-algebraic sets, whence also for semi-linear sets, this is equivalent to S being finite [11].[2]

The FO + linear expression

$$(\exists \vec{\varepsilon})(\forall \vec{x})(\forall \vec{y})(S(\vec{x}) \wedge S(\vec{y}) \Rightarrow -\vec{\varepsilon} < \vec{y} - \vec{x} < \vec{\varepsilon})$$

decides whether S is bounded.

The definitions of *topological interior*, *boundary*, and *closure* of a point-set in $\mathbf{R}^n$ can be translated almost straightforwardly into FO + linear using the property that the n-dimensional open *boxes*[3] form a basis for the standard topology of $\mathbf{R}^n$. This is shown in the following. The FO + linear expression

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

computes the topological interior of S. Similarly, the FO + linear expression

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

computes the topological closure of S. The topological boundary of S can be computed as the difference of the topological closure and the topological interior. Notice that the topological interior, boundary, and closure of a semi-linear set S of $\mathbf{R}^n$ is considered with respect to $\mathbf{R}^n$, and *not* with respect to the affine support of S. Later on in this chapter, we show that the affine support of S cannot be computed in FO + linear.

We note that Clementini et al. and Egenhofer et al. showed in a series of papers [17, 21, 22, 26] that a whole class of topological relationships in the two-dimensional plane, such as *disjoint*, *in*, *contained*, *overlap*, *touch*, *equal*, and *covered*, can be defined in terms of intersections between the boundary, interior, and complement of the geometric objects.

[2] It follows from this equivalence that the universal and existential quantor in the above formula may be swapped.

[3] An n-dimensional polytope S of $\mathbf{R}^n$ is an *n-dimensional box* if its bounding hyperplanes are parallel to the coordinate hyperplanes, i.e., there exist real numbers $l_1 \leq h_1, \ldots, l_n \leq h_n$ such that $S = \{(x_1, \ldots, x_n) \mid l_1 \leq x_1 \leq h_1 \wedge \ldots \wedge l_n \leq x_n \leq h_n\}$. A d-dimensional polytope of $\mathbf{R}^n$ of which all bounding hyperplanes, except the hyperplanes defining the affine support of that polytope, are parallel to the coordinate hyperplanes, is a *d-dimensional box*. An *open box* is the interior of a box with respect to its affine support.

Furthermore, the regularization of a semi-linear set, defined as the closure of its interior[4], can be computed in FO + linear, which is of importance, since the regularized set operators turn out to be indispensable in most geometric database applications [36, 51].

Notation 3.1 Since the queries above are so elementary, we will introduce predicates to denote them in later queries. We use $\mathtt{T}_{\overrightarrow{xy}}(S, \vec{z})$ as the predicate which decides whether $\vec{z}$ is a point of the translation of the semi-linear set S defined by the vector $\overrightarrow{xy}$. We denote by $\mathtt{symmetric}(S, \vec{p})$ the predicate which decides whether the semi-linear set S is point symmetric with respect to the point $\vec{p}$. Next, we use $\mathtt{finite}(S)$ and $\mathtt{bounded}(S)$ as predicates which decide whether the semi-linear set S is finite, respectively, bounded. Finally, $\mathtt{interior}(S, \vec{x})$, $\mathtt{closure}(S, \vec{x})$, and $\mathtt{boundary}(S, \vec{x})$ are predicates which decide whether $\vec{x}$ is a point in, respectively, the interior, the closure, and the boundary of the semi-linear set S. ∎

We next show that the Boolean query deciding whether a semi-linear set is *convex* can be defined in FO + linear.

Proposition 3.2 *The predicate* $\mathtt{convex}(S)$*, in which* S *is a semi-linear set of* $\mathbf{R}^n$*, and which evaluates to* true *if* S *is convex can be expressed in* FO + linear*.*

Proof. We prove that the FO + linear formula

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

defines the predicate $\mathtt{convex}(S)$.

In words, the above formula states that, for each pair of points in S, the midpoint is also in S. Clearly, if S is convex, then S satisfies the above FO + linear formula. Thus suppose, conversely, that S satisfies the above FO + linear formula. To prove that S is convex, we must show that, for arbitrary points $\vec{p}$ and $\vec{q}$ in S, the (open) line segment I connecting $\vec{p}$ and $\vec{q}$ is contained in S. Because of the property expressed by the above FO + linear formula, all points $k\vec{p} + (1-k)\vec{q}$, with k a dyadic number[5] between 0 and 1, are in S. Hence, $S \cap I$ is dense in I. Consequently, $I - S$, which is also a semi-linear set whence a semi-algebraic set, is zero-dimensional, and, therefore, finite [11]. A point in $I - S$, however, would then necessarily be the midpoint of (infinitely many) pairs of points of $S \cap I$, a contradiction. Thus, $I - S$ is empty, or I is contained in S. ∎

[4]Intuitively, a regular set has no dangling or isolated boundary points.

[5]A dyadic number is a finite sum of (positive and negative) integer powers of 2.

We continue with showing that it can be decided in FO + linear whether a given semi-linear set of $\mathbf{R}^n$ has a given number as its dimension. Since there are only finitely many values to consider for the dimension of a semi-linear set of $\mathbf{R}^n$, it follows that the dimension can actually be *computed* in FO + linear.

Definition 3.3 The dimension of a semi-linear set S of $\mathbf{R}^n$ is the maximum value of d for which there exists a d-dimensional neighborhood which is topologically open within its affine support and fully contained in S. The dimension of the empty set is defined as -1. ■

Again, without loss of generality, we can relax the condition "a d-dimensional neighborhood topologically open within its affine support" to "an open d-dimensional box."

The mathematical definition of dimension of a semi-linear set cannot be expressed straightforwardly in FO + linear, as in the case of the topological interior and closure of a semi-linear set. It is clear how to consider in FO + linear all d-dimensional boxes with an affine support parallel to a coordinate hyperplane, but, it is not clear how to consider all *arbitrary* d-dimensional boxes in FO + linear. So, we have to develop an alternative method to compute the dimension of a semi-linear set of $\mathbf{R}^n$ in FO + linear. First, we introduce the following notation.

Notation 3.4 Let S be a semi-linear set of $\mathbf{R}^n$. Then $\pi_i(S)$ denotes the semi-linear set

$$\{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \mid (\exists x_i) S(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)\}$$

of $\mathbf{R}^{n-1}$, i.e., the orthogonal projection of S onto the i-th coordinate hyperplane of $\mathbf{R}^n$. ■

We now show that, if $d < n$, at least one projection of S preserves the dimension. It is this observation that leads us to a method for computing the dimension of a semi-linear set in FO + linear.

Lemma 3.5 *If S is a d-dimensional semi-linear set of $\mathbf{R}^n$, and $d < n$, then there exists i, $1 \leq i \leq n$, such that the dimension of $\pi_i(S)$* equals *d.*

Proof. Since S has dimension d, there exists a d-dimensional box C, topologically open within its affine support, which is fully contained in S. Let $\vec{p}, \vec{r}_1, \ldots, \vec{r}_d$ be points in C such that the vectors $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$ are linearly independent. Since $d < n$, there exists i, $1 \leq i \leq n$, such that the unit vector $\vec{e}_i$ of the canonical coordinate system is not a linear combination of $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$. Consider $\pi_i(S)$. Clearly, $\pi_i(C)$

is convex and topologically open within its affine support, because C is convex and topologically open within its affine support. Let $\vec{q}, \vec{s}_1, \ldots, \vec{s}_d$ be the orthogonal projections on the ith coordinate hyperplane of $\vec{p}, \vec{r}_1, \ldots, \vec{r}_d$, respectively. We next show that $\overrightarrow{qs_1}, \ldots, \overrightarrow{qs_d}$ are linearly independent. Thereto, let $\lambda_1, \ldots, \lambda_d$ be real numbers such that $\lambda_1\overrightarrow{qs_1} + \cdots + \lambda_d\overrightarrow{qs_d} = \vec{0}$. Let $\vec{u}$ be the unique point of $\mathbf{R}^n$ for which $\overrightarrow{pu} = \lambda_1\overrightarrow{pr_1} + \cdots + \lambda_d\overrightarrow{pr_d}$. By the linearity of projection, $\pi_i(\overrightarrow{pu}) = \vec{0}$, whence $\overrightarrow{pu}$ is a multiple of $\vec{e}_i$, By choice of i, this multiple cannot be non-zero. Hence $\overrightarrow{pu} = \vec{0}$. From the linear independence of $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$, it then follows that $\lambda_1 = \cdots = \lambda_d = 0$. Thus $\overrightarrow{qs_1}, \ldots, \overrightarrow{qs_d}$ are linearly independent. Clearly, a convex set of $\mathbf{R}^n$, topologically open within its affine support, containing $d+1$ points $q, s_1, \ldots, s_d$ such that $\overrightarrow{qs_1}, \ldots, \overrightarrow{qs_d}$ are linearly independent, contains an open d-dimensional box. Since $\pi_i(S)$ cannot contain an open box of a strictly larger dimension, we have effectively shown that $\pi_i(S)$ is d-dimensional. ∎

We are now ready to claim that the dimension of a semi-linear set can be effectively computed in FO + linear.

Theorem 3.6 *The predicate* $\mathtt{dim}_n(S, d)$*, in which* S *is a semi-linear set of* $\mathbf{R}^n$ *and* d *is a number, and which evaluates to* true *if the dimension of* S *equals* d*, can be defined in* FO + linear*.*

Proof. The FO + linear formula defining the predicate $\mathtt{dim}_n(S, d)$ is obtained by induction on the dimension n. For convenience, we use in the following the predicates $\mathtt{empty}(S)$, $\mathtt{maxdim}(S)$, and $\mathtt{max}(d_1, \ldots, d_n, d)$ as macros for, respectively, the FO + linear expressions

1. $\neg(\exists\vec{x})S(\vec{x})$, which decides whether the semi-linear set S equals the empty set;
2. $(\exists\vec{x})(\exists\vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \wedge (\forall\vec{y})(\vec{x} - \vec{\varepsilon} < \vec{y} < \vec{x} + \vec{\varepsilon} \Rightarrow S(\vec{y})))$, which decides whether the semi-linear set S has maximal dimensional, i.e., the dimension of S equals n; and,
3. $(d = d_1 \vee \ldots \vee d = d_n) \wedge d \geq d_1 \wedge \ldots \wedge d \geq d_n$, which decides whether d is the maximum value of $d_1, \ldots, d_n$.

In one-dimensional space, the FO + linear formula

$$(d = -1 \wedge \mathtt{empty}(S)) \vee (d = 0 \wedge \neg\mathtt{empty}(S) \wedge \mathtt{finite}(S)) \vee (d = 1 \wedge \neg\mathtt{empty}(S) \wedge \neg\mathtt{finite}(S))$$

clearly defines $\texttt{dim}_1(S, d)$. Assume now that, in $\mathbf{R}^k$, $\texttt{dim}_k(S, d)$ has been defined in FO + linear, for $k < n$. Then, by the induction hypotheses and Lemma 3.5, the FO + linear formula

$$(d = n \wedge \texttt{maxdim}(S)) \vee (\neg\texttt{maxdim}(S) \wedge \texttt{dim}_{n-1}(\pi_1(S), d_1) \wedge \ldots \wedge \texttt{dim}_{n-1}(\pi_n(S), d_n) \wedge \texttt{max}(d_1, \ldots, d_n, d))$$

defines $\texttt{dim}_n(S, d)$ in $\mathbf{R}^n$. ■

In many situations, semi-linear sets contain parts of which the *local dimension* is strictly lower than the overall dimension. We show that we can select those parts of a semi-linear set which have a given local dimension. We first give a formal definition of the local dimension of a semi-linear set at one of its points.

Definition 3.7 Let S be a semi-linear set of $\mathbf{R}^n$, and let $0 \leq k \leq n$.

If $\vec{p}$ is a point of S, then S has *local dimension k at $\vec{p}$* if k is the maximum value of k such that, for each neighborhood V of $\vec{p}$ in $\mathbf{R}^n$, $S \cap V$ has dimension at least k.

The *k-dimensional component of S* is the set of all points of S in which S has local dimension k. ■

The different dimensional components of a semi-linear set yield a disjoint decomposition of the semi-linear set. The decomposition of a semi-linear set into its dimensional components is illustrated in Example 3.8.

Example 3.8 The two-dimensional component of the semi-linear set displayed in Figure 3.1 is made up of the filled triangle representing the hat and the square representing the nose. The line segments representing the mouth and the points on the edge of the face not belonging to the filled triangle constitute the one-dimensional component. Finally, the zero-dimensional component consists of the two points representing the eyes of the funny face. ■

Clearly, using the dimension predicate of Theorem 3.6, the corollary we present next follows immediately:

Corollary 3.9 *Let S be a semi-linear set of $\mathbf{R}^n$, let $\vec{p}$ be a point of S, and let $0 \leq k \leq n$.*

1. *The predicate $\texttt{localdim}_n(S, \vec{p}, k)$, which evaluates to* true *if the local dimension of S at $\vec{p}$ equals k, can be expressed in* FO + linear.

Figure 3.1: Illustration of the dimensional components of a semi-linear set.

2. *The query returning the k-dimensional component of S can be expressed in* FO + linear*.*

The decomposition of a semi-linear set into its dimensional components can be further refined using the notion of *regular stratification* of a semi-linear set [11, 46, 56, 75], which plays a key role in the study of the geometric structure of semi-linear sets.

We first study the notion of regular stratification in a more general context than semi-algebraic sets. Therefore, we introduce the following definition.

Definition 3.10 We define an **R**-*semi-algebraic set* as an n-dimensional point-set which can be described by a polynomial constraint formula in which the polynomials are allowed to have arbitrary *real* coefficients (instead of only real algebraic coefficients). Similarly, we define an **R**-*semi-linear set* as a point-set of $\mathbf{R}^n$ which can be described by a linear constraint formula in which the linear polynomials are allowed to have arbitrary *real* coefficients. ■

We now define *regular points* in the context of **R**-semi-algebraic sets and study the regular stratification of **R**-semi-algebraic sets.

Intuitively, a regular point of an **R**-semi-algebraic set is a point $\vec{p}$ of that set in which, locally, i.e., in some neighborhood of $\vec{p}$, the set looks like an algebraic variety (which can be described by equations only) which has a tangent space[6] in $\vec{p}$.

[6]A formal definition of the tangent space in a regular point of a set will be given after Definition 3.12. The notion is a generalization to arbitrary dimensions of tangent line to a curve and tangent plane to a surface.

Example 3.11 Consider the semi-algebraic set

$$S = \{(x, y) \mid x^2 + (y-1)^2 < 4 \vee x^2 + (y-1)^2 = 9 \vee (y = 1 \wedge 3 \leq x \leq 5) \vee (x = 5 \wedge y = 5)\}$$

in the plane, shown in Figure 3.2, which consists of a closed disk, a circle to which a closed line segment has been attached, and an isolated point.

Figure 3.2: The semi-algebraic set of Example 3.11.

The regular points of S are the points in the interior of the disk, the points of the circle with exception of the point $(3, 1)$, the points of the open line segment, and the isolated point. In a sufficiently small neighborhood of a point in the interior of the disk, the set S looks like the whole plane, which happens to be also the tangent space in that point. In a sufficiently small neighborhood of a point of the circle different from the point $(3, 1)$, the set S looks like the circle $x^2 + (y-1)^2 = 9$. The tangent space in that point is the corresponding tangent to the circle. In a sufficiently small neighborhood of a point in the open line segment, the set S looks like the line $y = 1$, which happens to be also the tangent space in that point. Finally, in a sufficiently small neighborhood of the isolated point, S looks like that isolated point, which happens to be its own tangent space. In any neighborhood of a point on the boundary of the disk or the other end point of the line segment, S does not look like an algebraic variety, and, therefore, these points are *not* regular. ■

We now formalize the intuition given above.

Definition 3.12 Let S be an $\mathbf{R}$-semi-algebraic set of $\mathbf{R}^n$, and let $\vec{p}$ be a point of S. The point $\vec{p}$ is a *regular point* of S if there exists a neighborhood V of $\vec{p}$ and

polynomials with real coefficients $P_1, \ldots, P_k$ in n real variables such that[7]

$$\frac{\mathrm{d}P_1}{\mathrm{d}\vec{x}}(\vec{p}), \ldots, \frac{\mathrm{d}P_k}{\mathrm{d}\vec{x}}(\vec{p})$$

are linearly independent and $S \cap V = \{\vec{x} \in V \mid P_1(\vec{x}) = \cdots = P_k(\vec{x}) = 0\}$. ■

If S and $\vec{p}$ satisfy the above conditions, then, locally around $\vec{p}$, S is an $(n-k)$-dimensional algebraic variety which has an $(n-k)$-dimensional tangent space at $\vec{p}$, defined by the system of k linear equations

$$\frac{\mathrm{d}P_1}{\mathrm{d}\vec{x}}(\vec{p}) \,.\, \vec{x} = 0, \ldots, \frac{\mathrm{d}P_k}{\mathrm{d}\vec{x}}(\vec{p}) \,.\, \vec{x} = 0.$$

Furthermore, the local dimension of S at $\vec{p}$ equals $n-k$, and the dimension of S is the maximum of all these numbers.

Now, let S be an **R**-semi-algebraic set, and let $Reg(S)$ be the set of those regular points of S in which S has local dimension precisely the dimension of S itself. It is well-known that the connected components of $Reg(S)$ are **R**-semi-algebraic [46, 56]. These are called *regular strata*. To $S - Reg(S)$, which is again **R**-semi-algebraic and of strictly lower dimension than S, we apply the same procedure, and continue until no more points are left. Since the dimension of the remaining set decreases strictly after each pass, this process is bound to stop. Upon completion of this process, every point of S is now part of one of the finitely many regular strata that have been generated, all of which are pairwise disjoint. We have thus constructed a finite partition called the *regular stratification* of S.

Example 3.13 We consider again the semi-algebraic set S of Example 3.11, shown in Figure 3.2. In Example 3.11, we observed that the set of regular points of S consists of the interior of the disk, the circle without the point $(3, 1)$, the open line segment, and the isolated point. Only in the points of the interior of the disk does S have dimension 2. They form the only stratum in the first layer of the regular stratification. The complement of the first layer with respect to S, say S^1, consists of the boundary of the disk, the wider circle, the closed line segment attached to it, and the isolated point, and has overall dimension 1. Its set of regular points consists of the boundary of the disk, the wider circle without the point $(3, 1)$, the open line segment, and the isolated point. Only in the regular points of the two circles and the line segment does S^1 have dimension 1. The boundary of the disk, the wider circle without the point $(3, 1)$, and the open line segment are therefore the 3 strata of the

[7] For a function $f : \mathbf{R}^n \to \mathbf{R}$, and a point $\vec{p}$ of $\mathbf{R}^n$, $\frac{\mathrm{d}f}{\mathrm{d}\vec{x}}(\vec{p})$, the *gradient* of f, also denoted as $\nabla f(\vec{p})$, is defined as $\left(\frac{\partial f}{\partial x_1}(\vec{p}), \ldots, \frac{\partial f}{\partial x_n}(\vec{p})\right)$.

second layer of the regular stratification. Each of the remaining 3 points constitute a stratum in the third and final layer of the regular stratification of S. Notice that not every point in a stratum is a regular point of the original set S. For example, the final layer consists of three points only one of which is a regular point of S. ■

We now review regular points and regular stratification in the context of **R**-semi-*linear* sets and show that the regular points of an **R**-semi-linear set, whence also of a **Z**-linear or **A**-linear set, can be computed within FO + linear.

If S is an **R**-semi-linear set, then, locally around a regular point $\vec{p}$, S will coincide with its tangent space at $\vec{p}$. This observation leads us to the following result.

Lemma 3.14 *Let S be an* **R***-semi-linear set of* $\mathbf{R}^n$*. The* FO + linear *expression*

$$S(\vec{x}) \wedge (\exists\vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \wedge (\forall\vec{y})(\forall\vec{z})(S(\vec{y}) \wedge \vec{x} - \vec{\varepsilon} \leq \vec{y} \leq \vec{x} + \vec{\varepsilon} \wedge S(\vec{z}) \wedge \\ \vec{x} - \vec{\varepsilon} \leq \vec{z} \leq \vec{x} + \vec{\varepsilon} \Rightarrow (\exists\vec{u})(2\vec{u} = \vec{y} + \vec{z} \wedge S(\vec{u})) \wedge (\exists\vec{v})(\vec{v} = 2\vec{x} - \vec{y} \wedge S(\vec{v}))))$$

defines the linear query returning the regular points of S.

Proof. Figure 3.3 illustrates what the above FO + linear expression means: a point $\vec{x}$ satisfies the expression if there exists a neighborhood V of $\vec{x}$ such that $S \cap V$ is closed under taking mid-points and symmetric with respect to $\vec{x}$.

Figure 3.3: Graphical illustration of the meaning of the FO + linear expression in Lemma 3.14.

If $\vec{x}$ is a regular point of S, then there exists a neighborhood V of $\vec{x}$ such that $S \cap V$ can be described by linear polynomial *equations*, whence S coincides with its affine support T within V. By choosing V convex and symmetric with respect to $\vec{x}$, it is immediately seen that $S \cap V = T \cap V$ is convex (whence closed under taking mid-points) and symmetric with respect to $\vec{x}$.

Conversely, let $\vec{x}$ be a point of S which has a neighborhood V such that $S \cap V$ is closed under taking mid-points and symmetric with respect to $\vec{x}$. From the proof

of Proposition 3.2, it follows immediately that S is convex, as S is closed under taking mid-points. Let T be the affine support of $S \cap V$, and suppose that the dimension of T equals k. Let $\vec{y}_1, \ldots, \vec{y}_k$ in $S \cap V$ such that $\overrightarrow{x\vec{y}}_1, \ldots, \overrightarrow{x\vec{y}}_k$ are linearly independent points. Since $S \cap V$ is convex and symmetric with respect to $\vec{x}$, the closed k-dimensional parallelopipedon P with mid-point $\vec{x}$ spanned by corner points $\vec{y}_1, \ldots, \vec{y}_k$ is fully contained within $S \cap V$. Clearly, P is a neighborhood of $\vec{x}$, and $P = P \cap T = P \cap S$. Hence, locally around $\vec{x}$, S coincides with T, whence $\vec{x}$ is a regular point of S. ∎

From Corollary 3.9 and Lemma 3.14, we obtain the following result.

Theorem 3.15 *Let S be an* $\mathbf{R}$*-semi-linear set of* $\mathbf{R}^n$*. The predicate* $\mathtt{reg}(S, \vec{x})$*, which evaluates to* true *if $\vec{x}$ is a regular point of S and the local dimension of S at $\vec{x}$ equals the overall dimension of S, can be expressed in* FO + linear*.*

Hence, the subsequent layers of regular points encountered during a regular stratification of a semi-linear set are again semi-linear (in any of the approaches considered) and can effectively be computed (if the set is semi-linear). Since the regular strata of a semi-linear set are the connected components of the regular layers of that set, it is unlikely that they can be computed in FO + poly.

We now give an example of the regular stratification of a semi-linear set.

Example 3.16 Consider the semi-linear set

$$\begin{aligned} S \;=\; & \{(x,y,z) \mid (0 \le x \le 3 \wedge 0 \le y \le 3 \wedge 0 \le z \le 3) \vee \\ & (3 \le x \le 5 \wedge y = 1 \wedge z = 0) \vee (x = 5 \wedge y = 5 \wedge z = 0)\} \end{aligned}$$

in three-dimensional space, shown in Figure 3.4, which consists of a closed filled cube to which a closed line segment has been attached, and an isolated point.

The set of regular points of S consists of the interior of the cube, the open line segment, and the isolated point. Only in the points in the interior of the cube does S have dimension 3. They form the only stratum in the first layer of the regular stratification. The complement of the first layer with respect to S, say S^1, consists of the faces of the cube, the closed line segment attached to it, and the isolated point, and has overall dimension 2. Its set of regular points consists of the 6 open faces of the cube, the open line segment, and the isolated point. Only in the points of the open faces does S^1 have dimension 2. The open faces of the cube are therefore the 6 strata of the second layer of the regular stratification. The complement of this second layer with respect to S^1, say S^2, consists of the edges of the cube, the line segment attached to it, and the isolated point, and has overall dimension 1. Its set

Figure 3.4: The semi-linear set of Example 3.16.

of regular points consists of the 12 open edges of the cube with the exception of the point $(3, 1, 0)$, the open line segment, and the isolated point. Only in the said points of the open edges and in the points of the open line segment does S^2 have dimension 1. The two open line segments in which the point $(3, 1, 0)$ divides one of the edges of the cube, the 11 remaining open edges, and the open line segment are therefore the 14 strata of the third layer of the regular stratification. The remaining 8 corner points, the end points of the line segment meeting the cube, and the isolated point each constitute one of the 11 strata in the fourth and final layer of the regular stratification of S. ■

Finally, we give two examples of queries that can be expressed in terms of the layers of the regular stratification of a semi-linear set.

Example 3.17 *Corners of a polygon.*

Let S be a closed filled polygon in the plane. Then $(S - \texttt{reg}(S)) - \texttt{reg}(S - \texttt{reg}(S))$ is the set of all corner points of this polygon. By our results, this set can be computed in FO + linear-**Z**. This technique can of course be generalized to higher-dimensional simplices and higher-dimensional spaces. ■

It must be noted that not only the zero-dimensional layer, but also the one-dimensional layer of the regular stratification of a semi-linear set is of interest.

Example 3.18 *Wire frame of a polyhedron.*

Let S be a closed filled polyhedron in three-dimensional space. Then $S - \texttt{reg}(S) - \texttt{reg}(S - \texttt{reg}(S))$ is the wire frame of S. As in Example 3.17, the wire frame can be computed in FO+linear-**Z**. Again, the computation of wire frames can be generalized to higher-dimensional polyhedra in higher-dimensional spaces. ■

3.2 Case Study: Linear Queries on Unions of Affine Subspaces

In this section, we illustrate the potential of the FO + linear-expressible queries given in the previous section. Our application domain is the class of queries on databases which consist of finite unions of affine subspaces of the n-dimensional Euclidean space, and, more particularly, we are interested in the expressiveness in FO + linear of the geometric notions *parallelism* and *orthogonality*.

We start with showing that, in FO + linear, it is possible to identify a semi-linear set which consists of a finite union of affine subspaces. Thereto, we first prove that it can be decided in FO + linear whether a semi-linear set is an affine subspace. After that, we generalize this result to finite unions of affine subspaces.

Proposition 3.19 *Let S be a semi-linear set of $\mathbf{R}^n$. The Boolean query deciding whether S is an affine subspace of $\mathbf{R}^n$ can be expressed in* FO + linear.

Proof. We show that S is an affine subspace of $\mathbf{R}^n$ if and only if S is symmetric with respect to each of its points; since point symmetry is expressible in FO + linear (see Section 3.1), the result then immediately follows.

Clearly, each affine subspace of $\mathbf{R}^n$ is symmetric with respect to each of its points.

Thus suppose, conversely, that S is symmetric with respect to each of its points. If S consists of a single point, then, clearly, S is an affine subspace of $\mathbf{R}^n$. Otherwise, let $\vec{p}$ and $\vec{q}$ be arbitrary but different points of S, and let L be the line through $\vec{p}$ and $\vec{q}$. To show that S is an affine subspace of $\mathbf{R}^n$, we must prove that L is fully contained in S.

Thereto, consider $S \cap L$, and assume that $S \cap L \neq L$. Since a semi-algebraic set, whence a fortiori a semi-linear set, can only contain a finite number of isolated points [11], $S \cap L$ must be the disjoint union of a finite number of isolated points, non-degenerated intervals, and half-lines. Since S (whence $S \cap L$) is symmetric with respect to each of its points, non-degenerated intervals and half-lines cannot occur in this disjoint union, whence $S \cap L$ is a finite set of (at least two) isolated points. Clearly, $S \cap L$ is *not* point-symmetric with respect to any of its two "outermost" points (which, more formally, are the end points of the interval obtained by taking the convex closure of $S \cap L$), a contradiction with our initial assumption. Thus, $S \cap L = L$, whence L is fully contained in S. ∎

By combining Proposition 3.19 with the expressibility of the dimension predicate (Theorem 3.6), we immediately obtain that the Boolean query deciding whether

a semi-linear set is a k-dimensional affine subspace, $0 \leq k \leq n$, is expressible in FO + linear.

From the proof of Proposition 3.19, we conclude that requiring a semi-linear set to be globally symmetric with respect to each of its points is a strong condition. It is, however, also possible to require a semi-linear set to be *locally* symmetric (i.e., within some neighborhood) with respect to each of its points, which leads to the next result.

Proposition 3.20 *Let S be a semi-linear set of $\mathbf{R}^n$. The Boolean query deciding whether S is a finite union of affine subspaces of $\mathbf{R}^n$ can be expressed in* FO + linear.

Proof. We show that S is a finite union of affine subspaces of $\mathbf{R}^n$ if and only if S is topologically closed and locally symmetric with respect to each of its points. The former can be decided within FO + linear (see Section 3.1), and the latter is *true* when S satisfies the FO + linear sentence

$$(\forall \vec{p})(\exists \vec{\varepsilon})(\varepsilon \neq \vec{0} \wedge (\forall \vec{x})(S(\vec{x}) \wedge \vec{p} - \vec{\varepsilon} < \vec{x} < \vec{p} + \vec{\varepsilon} \Rightarrow (\exists \vec{y})(\vec{y} = 2\vec{p} - \vec{x} \wedge S(\vec{y})))).$$

From this, the result then immediately follows.

Clearly, each finite union of affine subspaces of $\mathbf{R}^n$ is topologically closed and locally symmetric with respect to each of its points.

Thus suppose, conversely, that S is topologically closed and locally symmetric with respect to each of its points. Consider the regular stratification of S, which is a decomposition of S into a finite number of disjoint semi-linear sets. Each regular stratum is topologically open within its affine support. Clearly, the set S is a finite union of affine subspaces if, for each regular stratum of S, the affine support of that stratum is fully contained in S.

For the zero-dimensional regular strata of S, the above property is trivially satisfied. To see that the property also holds for higher dimensional strata, let $\vec{p}$ and $\vec{q}$ be arbitrary but different points in such a regular stratum, and let L be the line through $\vec{p}$ and $\vec{q}$. We prove that L is fully contained in S.

Thereto, consider $S \cap L$, and assume that $S \cap L \neq L$. Since S and L are both topologically closed, so is $S \cap L$. Thus, $S \cap L$ is the disjoint union of a finite number of isolated points, non-degenerated *closed* intervals, and *closed* half-lines. Moreover, $S \cap L$ *must* contain non-degenerated closed intervals and/or closed half-lines, since the stratum under consideration is non-zero-dimensional and open within its affine support. However, S cannot be locally symmetric with respect to the boundary points of these intervals and/or half lines, a contradiction with our initial assumption. Thus, $S \cap L = L$, whence L is fully contained in S.

Now, for any set which is topologically open within its affine support, that affine support is generated by the lines connecting different points of the set. Thus, we have indeed shown that the affine support of each regular stratum of S is fully contained in S. ■

Notice that the topological-closedness condition in Proposition 3.20 is essential, as local point symmetry alone is a much weaker property: for instance, each set which is topologically open within its affine support is locally symmetric with respect to each of its points. In the proof, the topological-closedness condition is used to ensure that boundary points with respect to which S would not be locally point-symmetric have to belong to S.

As for Proposition 3.19, combining Proposition 3.20 with the expressibility of the dimension predicate (Theorem 3.6) or the local dimension predicate (Corollary 3.9) yields the expressibility in FO + linear of several other Boolean queries.

We now turn to the issues of parallelism and orthogonality.

Proposition 3.21 *Let S be a k-dimensional affine subspace of $\mathbf{R}^n$, $0 \leq k \leq n$, and let $\vec{p}$ be a point of $\mathbf{R}^n$. The linear query returning the k-dimensional affine subspace of $\mathbf{R}^n$ through $\vec{p}$ parallel to S can be expressed in* FO + linear.

Proof. The FO + linear formula $(\exists \vec{y})(S(\vec{y}) \wedge \mathtt{T}_{\overrightarrow{yp}}(S, \vec{x})$ computes the query under consideration. ■

In order to prove a similar proposition for orthogonality, we need the following technical lemma.

Lemma 3.22 *Let S be a k-dimensional* linear *subspace of $\mathbf{R}^n$, $2 \leq k \leq n$.*

1. *The linear subspace S is spanned by its intersections with those $(n-1)$-dimensional coordinate hyperplanes in which S is not contained.*
2. *The linear subspace S is spanned by the lines that are intersections of S with one of the $(n-k+1)$-dimensional coordinate subspaces of $\mathbf{R}^n$.*

Proof.

1. The proof of the first statement of Lemma 3.22 goes by induction on n. If $n = 2$, then $k = 2$, and S equals the entire plane. The coordinate hyperplanes are the coordinate axes, and they equal their intersections with S. Obviously, the coordinate axes span the plane. For higher values of n, there are two

cases to consider. If S is contained in one of the coordinate hyperplanes, say H, then the first statement of Lemma 3.22 follows from applying the induction hypothesis to S within H, which is of dimension $n-1$. (Notice that the $(n-2)$-dimensional coordinate hyperplanes of H are obtained by intersecting H with each of the other coordinate hyperplanes of $\mathbf{R}^n$.) If S is contained in none of the coordinate hyperplanes, let $H_1, \ldots, H_n$ be the coordinate hyperplanes. Obviously, $S \cap H_1, \ldots, S \cap H_n$ are all linear subspaces of dimension $k-1$. Now suppose that $S \cap H_1 = \cdots = S \cap H_n$. Then $\bigcap_{i=1}^{n} S \cap H_i$, which is $(k-1)$-dimensional, would equal $S \cap \bigcap_{i=1}^{n} H_i = \{\vec{0}\}$, which is zero-dimensional, a contradiction, since $k \geq 2$. Thus, $S \cap H_1, \ldots, S \cap H_n$ are not all equal. Consequently, the dimension of the linear subspace of $\mathbf{R}^n$ spanned by $S \cap H_1 = \cdots = S \cap H_n$ is strictly greater than $k-1$, whence it must be k, whence that linear subspace must equal S.

2. The proof of the second statement of Lemma 3.22 goes by induction on k. If $k = 2$, then the second statement of Lemma 3.22 reduces to the first statement of Lemma 3.22. For higher values of k, we first consider the coordinate hyperplanes with which S has an intersection of dimension $k-1$ (these intersections span S, by the first statement of Lemma 3.22), and then apply the induction hypothesis to each of these intersections within their respective $(n-1)$-dimensional hyperplanes, yielding the desired result. ∎

We are now ready to prove the following result.

Proposition 3.23 *Let S be a k-dimensional affine subspace of $\mathbf{R}^n$, $0 \leq k \leq n$, and let $\vec{p}$ be a point of $\mathbf{R}^n$. The linear query returning the $(n-k)$-dimensional affine subspace through $\vec{p}$ orthogonal to S can be expressed in* FO + linear.

Proof. By Proposition 3.21, we may assume without loss of generality that S is a linear subspace, i.e., goes through $\vec{0}$, the origin of the canonical coordinate system of $\mathbf{R}^n$. Moreover, by the same proposition, it suffices to prove that the query returning the $(n-k)$-dimensional linear subspace orthogonal to S can be expressed in FO + linear. In other words, we have to show that, if S is a k-dimensional *linear* subspace of $\mathbf{R}^n$, the linear query returning the $(n-k)$-dimensional *linear* subspace $S^{\perp}$ of $\mathbf{R}^n$ orthogonal to S can be expressed in FO + linear.

By the expressibility of the dimension predicate (Theorem 3.6), we may divide the problem in cases according to the dimension of S.

If S is zero-dimensional (i.e., equals $\{\vec{0}\}$), then $S^{\perp}$ equals $\mathbf{R}^n$, which is the evaluation of $\{\vec{x} \mid \texttt{true}\}$.

If S is one-dimensional, i.e., a line through the origin $\vec{0}$ of the canonical coordinate system, we propose an FO + linear formula of the form

$$\varphi_1(\vec{x}) \vee \ldots \vee \varphi_n(\vec{x}).$$

We first explain how $\varphi_1(\vec{x})$ is obtained.

We start by checking whether S is contained in the first coordinate hyperplane, i.e., whether the first coordinate of all points of S equals 0, which can easily be done in FO + linear. If this is the case, we return the empty set as partial output corresponding to $\varphi_1(\vec{x})$. Otherwise, we compute the unique point $\vec{p} = (1, p_2, \ldots, p_n)$ of S with first coordinate 1, which, again, can easily be done in FO + linear. The real formula

$$x_1 + p_2 x_2 + \cdots + p_n x_n = 0$$

defines the set of all vectors $\vec{x} = (x_1, \ldots, x_n)$ orthogonal to $\vec{p}$, which constitute $S^\perp$. Unfortunately, the above formula is not linear. We can demonstrate, however, that it is possible to compute the above products in FO + linear. Let, for $j = 2, \ldots, n$, the predicate $S_{1j}(x, y)$ be an abbreviation for the FO + linear formula

$$(\exists \vec{x})(S(\vec{x}) \wedge x_1 = x \wedge x_j = y).$$

Hence, S_{1j} represents the projection of S on the $(1, j)$th coordinate plane. Hence, S_{1j} is the line through the origin $(0, 0)$ and the point $(1, p_j)$ in $\mathbf{R}^2$, and, therefore, $S_{1j}(x, y)$ is *true* if and only if $y = p_j x$. The real formula $x_1 + p_2 x_2 + \cdots + p_n x_n = 0$ is therefore equivalent to the FO + linear formula

$$(\exists y_2) \ldots (\exists y_n)(S_{12}(x_2, y_2) \wedge \ldots \wedge S_{1n}(x_n, y_n) \wedge x_1 + y_2 + \cdots + y_n = 0).$$

Hence, in the case considered, it is possible to return $S^\perp$ as partial output corresponding to $\varphi_1(\vec{x})$.

The construction of $\varphi_i(\vec{x})$, $2 \leq i \leq n$, is analogous, with the role of the first coordinate plane taken over by the ith coordinate plane.

Since S is not parallel to at least one of the coordinate hyperplanes, at least one of the partial outputs corresponding to $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x})$, respectively, is non-empty, whence the union of all the partial outputs is always $S^\perp$, which concludes the case that S is one-dimensional.

If S is of higher dimension, we first compute the intersections of S with the $(n-k+1)$-dimensional coordinate subspaces. Let E be such a subspace. If $S \cap E$ is one-dimensional (i.e., a line), which can be checked in FO + linear, then we return $(S \cap E)^\perp$ as partial output, computed as in the previous case. Otherwise, we return $\mathbf{R}^n$ as partial output. By Lemma 3.22, the *intersection* of all these partial outputs yields $S^\perp$. ∎

In the two-dimensional plane $\mathbf{R}^2$, Proposition 3.23 says that is possible to compute in FO + linear the line through a given point orthogonal to a given line, i.e., making a right angle with it. It is possible to generalize this result to other angles:

Corollary 3.24 *Let S, T_1 and T_2 be lines in the plane $\mathbf{R}^2$, and let $\vec{p}$ be a point of $\mathbf{R}^2$. The linear query returning the line through $\vec{p}$ making the same angle with S as T_2 with T_1 can be expressed in* FO + linear.

Proof. As explained at the beginning of the proof of Proposition 3.23, we may assume without loss of generality that all lines concerned go through $\vec{0}$, the origin of the canonical coordinate system, and that $\vec{p} = \vec{0}$. The geometric construction of the required line shown in Figure 3.5 demonstrates that Corollary 3.24 is indeed an immediate consequence of Proposition 3.23. ∎

Figure 3.5: Geometric construction of the line through $\vec{0}$ making the same angle with S as T_2 with T_1. First, the thin lines $S^\perp$ and ${T_1}^\perp$, orthogonal to S and T_1, respectively, are drawn. The construction starts with an arbitrary point of T_2 and yields a point of the result. Each point of the result can be reached in this way.

It can be seen that Corollary 3.24 still holds if S, T_1, T_2, and $\vec{p}$ are contained in a plane of a space $\mathbf{R}^n$ of arbitrary dimension. Indeed, if T_1 and T_2 are parallel (i.e., are equal or have no point in common), then the output of the query in Corollary 3.24 is the line through $\vec{p}$ parallel to S, which can be computed in FO + linear by Proposition 3.21. If T_1 and T_2 are not parallel, then $\vec{q}$ is in the plane supported by T_1 and T_2 if and only if $\vec{q}$ is the mid-point of two points in $T_1 \cup T_2$. Hence, this plane can be computed in FO + linear. Using this additional information and Proposition 3.23,

it is not difficult to see that the construction in the proof of Corollary 3.24 can still be carried out in FO + linear.

To close this section, we restrict ourselves to databases consisting of finite unions of one-dimensional affine subspaces, i.e., lines, and ascertain the expressibility in FO + linear of several linear queries pertaining to such sets.

We first summarize what we can determine in FO + linear about the composition of such sets (cfr. Proposition 3.19 and Proposition 3.20).

Proposition 3.25 *Let S be a semi-linear set of $\mathbf{R}^n$. The Boolean queries deciding the following properties can be expressed in* FO + linear*:*

1. *S consists of a finite number of lines;*
2. *S is a single line;*

Remark 3.26 Although the following Boolean queries are strictly spoken not within the context of databases consisting of unions of subspaces, we notice that they can be also expressed in FO + linear:

1. S consists of a finite number of lines, half-lines, and (non-degenerated) line segments only;
2. S is a single (open/closed) half-line;
3. S is a single (non-degenerated) (open/half-open-half-closed/closed) line segment.

The first query follows from the first part of the proof of Proposition 3.25.

For the second and third query, we first check that S consists of lines, half-lines, and (non-degenerated) line segments only (Query 1). If, in addition, S is convex (Theorem 3.2), then S is either a single line, or a single half-line, or a single (non-degenerated) line segment. Thus, if S is not bounded and S is not a line (Proposition 3.25), S is a half-line; if S is bounded, S is a non-degenerated interval. To decide which type of half-line or interval S is, we can in addition compute the set of points of S with respect to which S is *not* locally point-symmetric. These points are precisely the boundary points of S contained in S. The type of half-line or interval then depends on whether there are 0, 1, or 2 of them. ∎

We now focus on the case where S consists of a *finite number of lines* of $\mathbf{R}^n$.

Proposition 3.27 *Let S consist of a finite number of lines of $\mathbf{R}^n$, $n \geq 1$. The query returning the set of all intersection points between two or more lines in S can be expressed in* FO + linear.

Proof. The intersection points of two or more lines in S are precisely those points $\vec{p}$ of S in which S is not *locally convex*, i.e., those points $\vec{p}$ of S such that, for no convex neighborhood V of $\vec{p}$ in $\mathbf{R}^n$, $S \cap V$ is convex. Since the neighborhoods used in the topological queries earlier in this section are convex and suffice to express the above property, Proposition 3.27 now follows from Theorem 3.2. ∎

Corollary 3.28 *Let S be a semi-linear set of $\mathbf{R}^n$. The Boolean query deciding whether S consists of a finite number of parallel lines can be expressed in* FO+linear.

Proof. By Proposition 3.25, we can decide whether S consists of a finite number of lines; S will consist of a finite number of parallel lines if and only the set of intersection points between two or more lines in S is empty, which can be decided in FO + linear by Proposition 3.27. ∎

For the following result, we need a technical lemma.

Lemma 3.29 *Let S consists of k parallel lines of $\mathbf{R}^n$, $k \in \mathbf{N}$ and $k > 0$. There exists a linear query expressible in* FO + linear *that selects a single line from these.*

Proof. First, assume that S is *not* parallel to the ith coordinate hyperplane (i.e., the set of ith coordinates of S is *not* a singleton), $1 \leq i \leq n$, and let, for $j = 1, \ldots, n$, $j \neq i$, $\mathtt{min}_{ij}(S)$ be the semi-linear set defined by the FO + linear formula

$$S(\vec{x}) \wedge (\forall \vec{y})(S(\vec{y}) \wedge y_i = x_i \Rightarrow x_j \leq y_j).$$

Then $\mathtt{min}_{ij}(S)$ consists of those lines of S that are "leftmost" with respect to the jth coordinate axis. Thus,

$$\mathtt{min}_i(S) = \mathtt{min}_{in}(\mathtt{min}_{i(n-1)}(\ldots \mathtt{min}_{i1}(S) \cdots))$$

consists of a single line.

We next modify the definition of $\mathtt{min}_i$ such that $\mathtt{min}_i(S) = S$ if S is parallel to the ith coordinate plane, a condition which can easily be checked in FO + linear.

Then, in all cases,

$$\mathtt{min}(S) = \mathtt{min}_n(\mathtt{min}_{n-1}(\ldots \mathtt{min}_1(S) \cdots))$$

consists of a single line, since S is not parallel to at least one of the coordinate hyperplanes. ∎

Proposition 3.30 *Let S consists of a finite number of lines of $\mathbf{R}^n$, and let $\vec{p}$ be a point of S. The linear query returning the semi-linear set consisting of all lines of S through $\vec{p}$ can be expressed in* FO + linear*.*

Proof. First, assume that $\vec{p}$ is not an intersection point of two or more lines of S. Then, precisely one line of S goes through $\vec{p}$. Consider a neighborhood V of $\vec{p}$ in $\mathbf{R}^n$ such that $S \cap V$ is convex. (Such a neighborhood exists, and its existence can be asserted in FO + linear using the techniques employed to express topological queries and Theorem 3.2.) Necessarily, $S \cap V$ is a line segment. Let $S_{\vec{p}}$ the set of all points $\vec{x}$ of S which have a neighborhood W in $\mathbf{R}^n$ for which $\mathtt{T}_{\overrightarrow{xp}}(S \cap W) \subseteq S \cap V$. Clearly, there exists an FO + linear formula defining $S_{\vec{p}}$. Since translations preserve parallelism, $S_{\vec{p}}$ consists of the lines of S parallel to the line of S through $\vec{p}$, from which the intersection points with other lines have been omitted. Hence, the topological closure of $S_{\vec{p}}$, $\overline{S}_{\vec{p}}$, which can be computed in FO + linear, consists of all lines of S parallel to the line of S through $\vec{p}$. From this set, a single line can be selected in FO + linear, by Lemma 3.29. The output of the query is the line through $\vec{p}$ parallel to this single line, which can be computed in FO + linear, by Proposition 3.21.

Next, assume that $\vec{p}$ is an intersection point of two or more lines of S. Consider a neighborhood V of $\vec{p}$ in $\mathbf{R}^n$ such that $S \cap V$ is point-symmetric with respect to $\vec{p}$. (Such a neighborhood exists, and its existence can be asserted in FO + linear, as was shown earlier in the proof of Proposition 3.20.) For every point $\vec{q}$ in $S \cap V - \{\vec{p}\}$, $\vec{q}$ is *not* an intersection point of lines of S, whence the semi-linear set consisting of all lines of S through $\vec{q}$ can be computed in FO + linear, as shown in the first part of the proof. The output of the query is the union of all these sets. ∎

Proposition 3.30 has some remarkable corollaries.

Corollary 3.31 *Let S consist of a finite number of lines of $\mathbf{R}^n$, and let $\vec{p}$ and $\vec{q}$ be points on some line of S. The linear queries respectively returning the line through $\vec{p}$ and $\vec{q}$ and the closed line segment between $\vec{p}$ and $\vec{q}$ can be expressed in* FO + linear*.*

Proof. The expressibility in FO + linear of the first query is an immediate consequence of Proposition 3.30. To see that the second query is expressible in FO+linear, let $\vec{r}$ be the mid-point of $\vec{p}$ and $\vec{q}$. The points common to the line through $\vec{p}$ and $\vec{q}$ and all neighborhoods of $\vec{r}$ containing both $\vec{p}$ and $\vec{q}$ constitute the closed line segment between $\vec{p}$ and $\vec{q}$. ∎

Corollary 3.32 *Let S consist of a finite number of lines of $\mathbf{R}^n$. The linear queries respectively returning all pairs of parallel lines and all pairs of orthogonal lines of S (which can be seen as a query of type $[0, n] \to [0, 2n]$) can be expressed in* FO + linear*.*

Proof. Let $\vec{p}$ and $\vec{q}$ be points of S which are not intersection points of two or more lines of S. (This can be decided in FO + linear, by Proposition 3.27). By Proposition 3.30, it is possible to compute in FO + linear the unique lines of S going through $\vec{p}$ and $\vec{q}$, respectively. By Corollary 3.28, it now follows that the first query can be expressed in FO + linear. By Proposition 3.23, it is decidable whether two lines are orthogonal, whence also the second query can be expressed in FO + linear. ■

3.3 Limitations of FO + linear

From the results proposed in Section 3.1 and Section 3.2, the reader may have the impression that FO + linear is a rather expressive query language and well-suited as a linear query language. In this section, however, we demonstrate that there exist fundamental linear queries not expressible in FO + linear.

In the literature (e.g., [2, 3, 4, 73]), researchers have been concerned with the non-definability of certain geometric sets by linear first-order formulae (i.e., with the non-semi-linearity of these sets). In this section, however, we are concerned with the non-expressibility of *queries* in FO + linear, as opposed to the non-definability of sets by linear formulae. The principal contribution of this section is the development of a general tool to lift the non-definability of certain semi-algebraic sets by linear formulae to the non-expressibility in FO + linear of closely related FO + poly queries. Application of this tool allows us to establish the non-expressibility in FO + linear of a wide range of queries in FO + poly^{lin}, which in turn improves our insight into the nature of the FO + poly^{lin} queries not expressible in FO + linear.

To develop this tool, we first establish a link between certain semi-algebraic sets and certain FO + poly queries.

Definition 3.33 Let P be a semi-algebraic subset of $(\mathbf{R}^n)^m$, $m, n \geq 1$. Let k be such that $0 \leq k \leq m$. Furthermore, assume that P and k are such that, for each sequence $\vec{u}_1, \ldots, \vec{u}_k$ in $\mathbf{R}^n$, and for all sequences $i_1, \ldots, i_k$ such that $\{\vec{u}_{i_1}, \ldots, \vec{u}_{i_k}\} = \{\vec{u}_1, \ldots, \vec{u}_k\}$, the following permutation invariance property holds, for $\vec{u}_{k+1}, \ldots, \vec{u}_m$ in $\mathbf{R}^n$:

$$(\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P \Leftrightarrow (\vec{u}_{i_1}, \ldots, \vec{u}_{i_k}, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P.$$

The query $Q_{P,k}$ of signature $[0, n] \to [0, n(m-k)]$ is now defined as follows. If S consists of at most k points of $\mathbf{R}^n$, say $S = \{\vec{u}_1, \ldots, \vec{u}_k\}$ ($\vec{u}_1, \ldots, \vec{u}_k$ not necessarily all distinct), then

$$Q_{P,k}(S) = \{(\vec{u}_{k+1}, \ldots, \vec{u}_m) \mid (\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P\};$$

otherwise $Q_{P,k}(S)$ is empty. ■

Observe that the invariance property assumed for P and k guarantees that $Q_{P,k}$ is a well-defined query expressible in FO + poly.

Example 3.34 We give some examples of sets P and corresponding queries $Q_{P,k}$ which will be used further on in this section.

1. Let P_1 be the set

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_1, \ldots, \vec{u}_m \text{ are collinear}\},$$

for appropriately chosen values of n and m. The set P_1 is obviously semi-algebraic; e.g., for $m = 3$, it is expressed by the real formula

$$\vec{u}_2 = \vec{u}_3 \vee (\exists\lambda_1)(\exists\lambda_2)(\lambda_1 + \lambda_2 = 1 \wedge \vec{u}_1 = \lambda_1\vec{u}_2 + \lambda_2\vec{u}_3).$$

Moreover, it satisfies Definition 3.33 for $k = m$. The associated query $Q_{P_1,m}$ can be interpreted as the Boolean query which decides whether a semi-linear set S consists of at most m collinear points.

2. Let P_2 be the set

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is in the same (closed) Voronoi cell as } \vec{u}_{m-1} \text{ with respect to } \vec{u}_1, \ldots, \vec{u}_{m-2}\}.$$

The point $\vec{u}_m$ belongs to the same (closed) Voronoi cell as $\vec{u}_{m-1}$ with respect to $\vec{u}_1, \ldots, \vec{u}_{m-2}$ if the condition

$$(\exists\vec{u})((\vec{u} = \vec{u}_1 \vee \ldots \vee \vec{u} = \vec{u}_{m-2}) \wedge \bigwedge_{i=1}^{m-2} (d(\vec{u}_{m-1}, \vec{u}) \leq d(\vec{u}_{m-1}, \vec{u}_i) \wedge d(\vec{u}_m, \vec{u}) \leq d(\vec{u}_m, \vec{u}_i))),$$

where $d(\vec{r}, \vec{s})$ denotes the Euclidean distance between $\vec{r}$ and $\vec{s}$, is satisfied. Hence, P_2 is semi-algebraic. Moreover, it satisfies Definition 3.33 for $k = m-2$. The associated query $Q_{P_2,m-2}$ of type $[0, n] \to [0, 2n]$ can be interpreted as the linear query that associates, with each semi-linear set S consisting of at most $m-2$ points, pairs of points which belong to the same Voronoi cell with respect to the points of S, and, with every other semi-linear set S, the empty set.

3. Let S be a semi-algebraic set of $\mathbf{R}^n$. We define the *diameter* of S, denoted $\emptyset(S)$, as the supremum[8] of the (Euclidean) distance between two points of S. Let
$$P_3 = \{(\vec{u}_1, \ldots, \vec{u}_{m-1}, (d, \underbrace{0, \ldots, 0}_{n-1})) \mid \emptyset(\{\vec{u}_1, \ldots, \vec{u}_{m-1}\}) = d\}.$$
It is easily seen that P_3 is a semi-algebraic set; e.g., for $m = 3$, it computes the distance between two points. Moreover, it satisfies Definition 3.33 for $k = m - 1$. The associated query $Q_{P_3,m-1}$ can be interpreted as the aggregate query of type $[0, n] \to [0, n]$ which associates, with each semi-linear set S consisting of at most $m - 1$ points, the singleton $(\emptyset(S), \underbrace{0, \ldots, 0}_{n-1})$. ■

We now establish that the query $Q_{P,k}$ is not expressible in FO + linear as soon as the set P is not definable by a linear constraint formula.

Theorem 3.35 *Let P be a semi-algebraic subset of $(\mathbf{R}^n)^m$, $m, n \geq 1$, let k be such that $0 \leq k \leq m$, and let P and k satisfy the conditions of Definition 3.33. If P is not definable by a linear constraint formula, then the following holds:*

1. *the query $Q_{P,k}$ is not expressible in* FO + linear*;*
2. *if Q is a linear query of type $[0, n] \to [0, n(m - k)]$ such that, for every semi-linear set S of $\mathbf{R}^n$, $Q(S) = Q_{P,k}(S)$ if $Q_{P,k}(S)$ is not empty, then Q is not expressible in* FO + linear*.*

Proof.

1. Assume, to the contrary, that the query $Q_{P,k}$ is expressible in FO + linear. Then there exists an FO + linear formula $\varphi_{P,k}(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)$, with R an appropriate predicate name, such that, for each semi-linear set S of $\mathbf{R}^n$, $Q_{P,k}(S) = \{(\vec{u}_{k+1}, \ldots, \vec{u}_m) \mid \varphi_{P,k}(S; \vec{u}_{k+1}, \ldots, \vec{u}_m)\}$. We now argue that the predicate name R must effectively occur in $\varphi_{P,k}$. If this were not the case, then the query associated with $\varphi_{P,k}$ would be independent of the input, i.e., a constant function. This constant function must return the empty set, since $Q_{P,k}$ by definition returns the empty set on all inputs containing more than k points. However, $Q_{P,k}$ cannot return the empty set on *every* input unless P is the empty set, which is obviously definable by a linear constraint formula, contrary the hypothesis of the theorem. Thus R must occur in $\varphi_{P,k}$.

[8]The supremum d of a set $S \subseteq \mathbf{R}$ is defined by $(\forall x)(S(x) \Rightarrow x \leq d) \wedge (\forall \varepsilon)(\varepsilon > 0 \Rightarrow (\exists x)(S(x) \wedge x > d - \varepsilon)$, which is obviously expressible in FO + linear.

Given the formula $\varphi_{P,k}$, we construct a new formula $\hat{\varphi}_{P,k}$, as follows. Let $\vec{x}_1, \ldots, \vec{x}_k$ be variables that do not occur in $\varphi_{P,k}$. Now replace every literal of the form

$$R(\vec{z})$$

in $\varphi_{P,k}$ by the formula

$$\vec{z} = \vec{x}_1 \vee \cdots \vee \vec{z} = \vec{x}_k.$$

Observe that the formula $\hat{\varphi}_{P,k}$ is a linear constraint formula with free variables $\vec{x}_1, \ldots, \vec{x}_m$. Our claim is that the formula $\hat{\varphi}_{P,k}$ defines the set P, a contradiction with the hypothesis of the theorem. To substantiate our claim, we consider an m-tuple $(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m$. From the definition of $Q_{P,k}$ and $\varphi_{P,k}$, we have

$$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow (\vec{u}_{k+1}, \ldots, \vec{u}_m) \in Q_{P,k}(\{\vec{u}_1, \ldots, \vec{u}_k\}),$$

whence

$$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow \varphi_{P,k}(\{\vec{u}_1, \ldots, \vec{u}_k\}; \vec{u}_{k+1}, \ldots, \vec{u}_m).$$

It follows from the construction of $\hat{\varphi}_{P,k}$ from $\varphi_{P,k}$ that

$$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow \hat{\varphi}_{P,k}(\vec{u}_1, \ldots, \vec{u}_m).$$

2. Assume that Q is expressible in FO + linear. Then there exists a formula

$$\varphi_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)$$

that defines Q, where R stands for the input predicate. Given φ_Q, we can construct the formula $\hat{\varphi}_Q$:

$$\hat{\varphi}_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m) \Leftrightarrow (|R| \leq k \wedge \varphi_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)) \vee (|R| > k \wedge \mathit{false}).$$

It is obvious that this expression for $\hat{\varphi}_Q$ can be translated into proper FO + linear syntax. It now follows from the properties of Q that the FO + linear-formula $\hat{\varphi}_Q$ expresses the query $Q_{P,k}$, which is impossible by the first part of the theorem. ∎

A shortcoming of Theorem 3.35 is that we have to find out somehow whether the semi-algebraic set P is semi-linear or not. Fortunately, in the following chapter, we will prove that it is *decidable* whether a semi-algebraic set is semi-linear. To apply Theorem 3.35 on the semi-algebraic sets of Example 3.34 without relying on the semi-linearity test of the following chapter, we show that these semi-algebraic sets are not definable by linear formulae for most values of m and n, by a reduction argument.

Proposition 3.36 *The sets P_1, P_2, P_3 are not definable by linear formulae if $n \geq 2$ and $m \geq 3$.*

Proof.

1. We first show that P_1 is not definable by a linear constraint formula. Assume to the contrary that P_1 is definable by a linear constraint formula for some $n \geq 2$ and some $m \geq 3$. Then, clearly, P_1 is also definable by a linear constraint formula for $n = 2$ and $m = 3$. Let $\texttt{collinear}(x_1, x_2, y_1, y_2, z_1, z_2)$ denote this formula. We now show that there exists a linear constraint formula $\texttt{product}(x, y, z)$, with x, y, z real variables, equivalent to the real formula $z = xy$, an obvious contradiction. From the geometric construction of the product shown in Figure 3.6, it follows that

$$(x = 0 \wedge z = 0) \vee (y = 0 \wedge z = 0) \vee (y = 1 \wedge z = x) \vee \\ \neg(\exists v)(\exists w)(\texttt{collinear}(x, 0, 0, 1, v, w) \wedge \texttt{collinear}(z, 0, 0, y, v, w))$$

   is the desired linear constraint formula.

Figure 3.6: Geometric construction of the product.

2. The semi-algebraic set P_2 is not definable by a linear formula, since P_1 is not: indeed, for $m = 4$, we have that

$$(\vec{p}_1, \vec{p}_2, \vec{p}_3) \in P_1 \Leftrightarrow (\exists \vec{p})(\exists \vec{q})((\vec{p}, \vec{q}, \vec{p}, \vec{p}_1) \in P_2 \wedge (\vec{p}, \vec{q}, \vec{p}, \vec{p}_2) \in P_2 \wedge \\ (\vec{p}, \vec{q}, \vec{p}, \vec{p}_3) \in P_2 \wedge (\vec{p}, \vec{q}, \vec{q}, \vec{p}_1) \in P_2 \wedge (\vec{p}, \vec{q}, \vec{q}, \vec{p}_2) \in P_2 \wedge (\vec{p}, \vec{q}, \vec{q}, \vec{p}_3) \in P_2).$$

3. The semi-algebraic set P_3 is not definable by a linear constraint formula, because, for $m \geq 3$, it is possible to obtain a disk by applying appropriate FO + linear-expressible operations to P_3. ∎

Theorem 3.35 and Proposition 3.36 yield the following corollary, the proof of which is immediate from the former:

Theorem 3.37 *1. The Boolean query of type $[0,n] \to [0,0]$ deciding whether a semi-linear set of $\mathbf{R}^n$ is contained in a line is not expressible in* FO + linear.

2. The linear query of type $[0,n] \to [0,2n]$ computing all pairs of points of $\mathbf{R}^n$ which belong to the same Voronoi cell with respect to a finite semi-linear set of $\mathbf{R}^n$ is not expressible in FO + linear.

3. The linear aggregate query of type $[0,n] \to [0,n]$ computing the singleton

$$\{(\emptyset(S), \underbrace{0, \ldots, 0}_{n-1})\}$$

upon a semi-linear set S of $\mathbf{R}^n$ as input, is not expressible in FO + linear *(whence the diameter query of type $[0,n] \to [0,1]$ is not expressible either).*

Obviously, Theorem 3.35 can be used to show the non-expressibility of many more linear queries. For instance, it can be used to prove Proposition 2.14 as well as the non-expressibility in FO + linear of several other Boolean queries. Just as Boolean queries restricted to semi-linear sets are necessarily linear, rational-valued aggregate queries, such as volume or surface restricted to semi-linear sets, are necessarily linear.

As a final example, we discuss the non-expressibility in FO + linear of the queries with type $[0,n] \to [0,n]$ which compute the convex closure and the affine support of a semi-linear set. We can show that for $n \geq 2$ and $m \geq 3$ the semi-algebraic sets

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is in the convex closure of } \{\vec{u}_1, \ldots, \vec{u}_{m-1}\}\}$$

and

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is in the affine support of } \{\vec{u}_1, \ldots, \vec{u}_{m-1}\}\}$$

cannot be expressed by linear formulae as the product of two real numbers would become expressible (in the same way as the definability of the set P_1 led to the expressibility of the product of two real numbers). Then, according to Theorem 3.35, we can lift the non-definability of these sets by linear formulae to the non-expressibility in FO + linear of the queries computing the convex closure and the affine support of a semi-linear set. For these last two queries, however, the non-expressibility can also be established more directly by reduction to the non-expressibility of the collinearity query.

3.4 Expressibility in FO + linear is Undecidable

Until now, we have studied the expressibility of FO+poly^{lin} queries within FO+linear for each FO + poly^{lin} query individually. In Section 3.1 and Section 3.2, we have proved that various FO + poly^{lin} queries are expressible within FO + linear. In Section 3.3, we provided a helpful tool to study the non-expressibility of FO+poly^{lin} queries within FO + linear. The most straightforward way to avoid this "query-by-query" study is to discover an algorithm to decide whether an FO + poly formula defining an FO + poly^{lin} query is expressible in FO + linear.

However, we must point out that we can prove the following theorem, which can be viewed as an analogue of Rice's Theorem for FO + poly-expressible queries:

Theorem 3.38 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be subclasses of polynomial constraint databases such that $\mathcal{C}_1$ contains all semantically finite geometric databases. Let* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *be the sublanguage of* FO + poly *consisting of those formulae that return outputs in $\mathcal{C}_2$ upon inputs in $\mathcal{C}_1$. Let $\mathcal{E}$ be a semantic property of* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formulae satisfying the following conditions:*

1. *if $\varphi(R_1, \ldots, R_k; x_1, \ldots, x_n)$ is an* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formula satisfying property $\mathcal{E}$ and defining a query of type, say, $[0, n_1; \ldots; 0, n_k] \to [0, n]$, and if $\vartheta(x_1, \ldots, x_{n_1})$ is a real formula defining a semantically finite geometric database relation of type $[0, n_1]$, then the* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formula*

$$\varphi'(R_2, \ldots, R_k; x_1, \ldots, x_n),$$

obtained from φ by substituting each occurrence of R_1 by ϑ, and defining a query of type $[0, n_2; \ldots; 0, n_k] \to [0, n]$, satisfies property $\mathcal{E}$;

2. *if $\varphi(R_2, \ldots, R_k; x_1, \ldots, x_n)$ is an* FO+poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formula satisfying property $\mathcal{E}$ and defining a query of type, say, $[0, n_2; \ldots; 0, n_k] \to [0, n]$, then, for each relation type $[0, n_1]$, and for each relation name R_1 of type $[0, n_1]$, the* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formula*

$$\varphi'(R_1, \ldots, R_k; x_1, \ldots, x_n) \equiv \varphi(R_2, \ldots, R_k; x_1, \ldots, x_n)$$

defining a query of type $[0, n_1; \ldots; 0, n_k] \to [0, n]$ satisfies property $\mathcal{E}$; and

3. *for some query type $[0, n_1; \ldots; 0, n_k] \to [0, n]$, there exist* FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formulae*

   - $\varphi^+(R_1, \ldots, R_k; x_1, \ldots, x_n)$
   - $\varphi^-(R_1, \ldots, R_k; x_1, \ldots, x_n)$

both defining a query of type $[0, n_1; \ldots; 0, n_k] \to [0, n]$ *such that* φ^+ *has property* $\mathcal{E}$ *and* φ^- *does not have property* $\mathcal{E}$.

Then it is undecidable whether an $\mathrm{FO} + \mathrm{poly}^{\mathcal{C}_1 \to \mathcal{C}_2}$ *formula has property* $\mathcal{E}$.

Proof. The proof is a variation of a proof of Paredaens, Van den Bussche, and Van Gucht [61] concerning undecidability of genericity in FO + poly (Theorem 1, p. 285).

The $\forall^*$-fragment of number theory is undecidable since Hilbert's 10th problem can be reduced to it. We encode a natural number n by the finite set $\mathtt{enc}(n) := \{0, \ldots, n\}$, and we encode a vector of natural numbers $(n_1, \ldots, n_k)$ by[9]

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

The corresponding decoding is first-order-expressible. Consider the $\mathrm{FO} + \mathrm{poly}^{\mathcal{C}_1 \to \mathcal{C}_2}$ formulae

$$\varphi^+(R_1, \ldots, R_k; x_1, \ldots, x_n)$$

and

$$\varphi^-(R_1, \ldots, R_k; x_1, \ldots, x_n)$$

defining queries of some common type, say, $[0, n_1; \ldots; 0, n_k] \to [0, n]$, such that φ^+ has property $\mathcal{E}$ and φ^- does not have property $\mathcal{E}$. We then reduce a $\forall^*$-sentence $(\forall \vec{x})\psi(\vec{x})$ of number theory to the following query of type $[0, 1; 0, n_1; \ldots; 0, n_k] \to [0, n]$ (S of type $[0, 1]$ and $R_1, \ldots, R_k$ of types $[0, n_1], \ldots, [0, n_k]$, respectively, are the input relation names of this query):

if (S encodes a vector $\vec{x}$) **then**
  if $\psi(\vec{x})$ **then**
    return$(\{(x_1, \ldots, x_n) \mid \varphi^+(R_1, \ldots, R_k; x_1, \ldots, x_n)\})$
  else
    return$(\{(x_1, \ldots, x_n) \mid \varphi^-(R_1, \ldots, R_k; x_1, \ldots, x_n)\})$
else
  return$(\{(x_1, \ldots, x_n) \mid \varphi^+(R_1, \ldots, R_k; x_1, \ldots, x_n)\})$.

By definition, the above query is $\mathrm{FO} + \mathrm{poly}^{\mathcal{C}_1 \to \mathcal{C}_2}$-expressible. Moreover, an $\mathrm{FO} + \mathrm{poly}^{\mathcal{C}_1 \to \mathcal{C}_2}$ formula computing this query can effectively be constructed. Let

$$\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$$

be such an formula. When the $\forall^*$-sentence $(\forall \vec{x})\psi(\vec{x})$ is valid, then

$$\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n) \equiv \varphi^+(R_1, \ldots, R_k; x_1, \ldots, x_n)$$

[9] For N a set of natural numbers and n a natural number, $N + n$ denotes the set $\{x + n \mid x \in N\}$.

has property $\mathcal{E}$, by conditions 2 and 3 above. When the $\forall^*$ sentence $(\forall\vec{x})\psi(\vec{x})$ is not valid, then $\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$ does not have property $\mathcal{E}$. To see this, let $\vec{n}$ be a vector of natural numbers for which $\psi(\vec{n})$ is *false*, and let $\vartheta(x)$ be a real formula defining $\mathtt{enc}(\vec{n})$. Let

$$\varphi'(R_1, \ldots, R_k; x_1, \ldots, x_n)$$

be the FO + poly$^{\mathcal{C}_1 \to \mathcal{C}_2}$ formula obtained from $\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$ by substituting each occurrence of S by ϑ. Then, clearly, φ' defines a query of type $[0, n_1; \ldots; 0, n_k] \to [0, n]$. Now, if $\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$ would have property $\mathcal{E}$, then $\varphi'(R_1, \ldots, R_k; x_1, \ldots, x_n)$ would have property $\mathcal{E}$, by condition 1, a contradiction with condition 3, since

$$\varphi'(R_1, \ldots, R_k; x_1, \ldots, x_n) \equiv \varphi^-(R_1, \ldots, R_k; x_1, \ldots, x_n).$$

Thus, clearly, $\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$ does *not* satisfy property $\mathcal{E}$. In summary, $\varphi(S, R_1, \ldots, R_k; x_1, \ldots, x_n)$ has property $\mathcal{E}$ if and only the $\forall^*$-sentence $(\forall\vec{x})\varphi(\vec{x})$ is valid. ■

From the proof of Theorem 2.14, we see that, without loss of generality, the real formula ϑ in condition 1 can be assumed to be **Z**-linear.

If we combine Proposition 2.14 and Theorem 3.38, in which we let $\mathcal{C}_1 = \mathcal{C}_2$ be the class of linear constraint databases, and $\mathcal{E}$ be FO + linear-expressibility, then we immediately obtain the following corollary:

Corollary 3.39 *It is undecidable whether an* FO + poly^{lin} *query induced by an* FO + poly *formula can be expressed in* FO + linear*.*

Hence, an algorithm to decide whether an FO + poly expression defining an FO + poly^{lin} query is expressible in FO + linear, cannot be provided.

The following results show that, even for subclasses of the linear constraint databases, it remains undecidable whether the corresponding FO + poly^{lin} queries are expressible in FO + linear.

Let $\mathcal{C}_1$ be the class of *finite* linear constraint databases, i.e., those semi-linear sets S on which the predicate $\mathtt{finite}(S)$ yields *true*. Denote $\mathcal{C}_2$ the class of arbitrary linear constraint databases. Again, let $\mathcal{E}$ be FO + linear-expressibility. From Section 3.3, we remember that the query which computes the diameter of a finite set of points cannot be expressed in FO+linear. Using Theorem 3.38, we then obtain immediately the following result:

Corollary 3.40 *It is undecidable whether an* FO+poly^{lin} *query which returns semi-linear outputs upon finite semi-linear inputs is expressible in* FO + linear*.*

This result remains true for the FO + poly^{lin} queries that return *finite* semi-linear outputs upon finite semi-linear inputs, i.e., the class of FO + poly^{lin} queries that preserve the classical notion of safety. In Section 5.5, we show that we can "lift" a query language complete for the FO + poly queries that return finite outputs upon finite inputs to a query language complete for the FO + poly^{lin} queries. The most straightforward way to obtain a query language complete for the FO + poly queries that return finite outputs upon finite inputs is to discover an algorithm to decide whether an FO + poly query returns finite outputs upon finite inputs. From the above result, it follows that such an algorithm does not exists.

We can generalize the above result to the class of databases which contain only k-dimensional affine spaces[10], $k > 0$. In this case, we study the decidability of FO + linear-expressibility of those FO + poly^{lin} queries which return semi-linear outputs upon input databases containing a finite number of k-dimensional affine spaces. Again, without going into details, we can prove that it is undecidable whether such an FO + poly^{lin} query can be expressed in FO + linear.

[10]Points can be seen as zero-dimensional affine spaces.

Chapter 4

Decidability of Semi-Linearity of Semi-Algebraic Sets

In this chapter, we focus on the decidability of semi-linearity for semi-algebraic sets. This is by no means a trivial problem, since it is possible to describe semi-linear sets by non-linear constraint formulae, as is illustrated in Example 2.11. Moreover, as explained in Section 2.3, there are two natural ways to define semi-linearity, **A**-semi-linearity and **Z**-semi-linearity, depending on the type of coefficients (real algebraic versus integer) of the linear inequalities used to describe a semi-linear set.

We first prove that semi-linearity for semi-algebraic sets is decidable in the **A**-semi-linear case, and we provide an FO + poly expression for the corresponding decision query. Next, we prove that semi-linearity for semi-algebraic sets is decidable in the **Z**-semi-linear case, too. However, we also prove that, in this case, there is *no* FO + poly expression for the corresponding decision query.

In the proof of the decidability results, the notion of regular stratification introduced in Section 3.1, plays a key role. More concretely, we propose an algorithm based on regular stratification to decompose a semi-linear set. We then study relevant properties of the so-obtained *algorithmic decomposition* of a semi-linear set, and use these to develop an algorithm, expressible in FO + linear, that computes the set of so-called *special points* of a semi-linear set. We show that a collection of hyperplanes defining every affine support of a subset of the special points of a semi-linear set defines a decomposition of that semi-linear set into convex cells.

We wish to point out that the road we chose to prove the decidability results in

this chapter is by no means the only one possible. An anonymous referee of the extended abstract of the paper [20], presented at the 16th *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, kindly suggested an alternative method based on decidability results in logic. For completeness, an outline of this method is given in Appendix B. Although that proof is much shorter, we stress that our approach yields some interesting by-products, among which a finite encoding technique for arbitrary semi-linear sets that is the basis of the design of more expressive query languages than FO + linear, which we study in Chapter 5.

An outline of this chapter is as follows. In Section 4.1, we recall the definitions of **R**-semi-algebraic and **R**-semi-linear sets given in Section 3.1. We then propose Property SL and show that **R**-semi-algebraic sets satisfying Property SL are necessarily **R**-semi-linear. The proof uses a decomposition of an **R**-semi-linear set based on the notion of regular stratification, which we shall refer to as the *canonical decomposition*. Unfortunately, it is unlikely that the techniques used in that proof can be expressed in FO + poly, as they involve connectivity [9, 33]. In Section 4.2, we propose an algorithm which uses a variation on those techniques, but does not require the expressibility of connectivity. This algorithm is expressible in FO + poly. We show that the resulting *algorithmic decomposition* of a semi-linear set has the same desirable properties as the decomposition used in the proof that **R**-semi-algebraic sets satisfying Property SL are necessarily **R**-semi-linear. In Section 4.3, algorithmic decompositions are used to show that **R**-semi-linear semi-algebraic sets are **A**-semi-linear, as a consequence of which Property SL (which can easily be expressed in FO + poly) can be used to decide whether a semi-algebraic set defined by a real constraint formula is **A**-semi-linear. To obtain this result, the bounded case is considered first, which is then bootstrapped to the general case. Finally, in Section 4.4, the technique developed in Section 4.3 is reviewed to obtain that **Z**-semi-linearity is decidable. However, we also show that **Z**-semi-linearity can*not* be decided by an FO + poly formula.

4.1 Property SL

In Chapter 2, we started with formulae built from polynomial equations and inequalities with *real algebraic* coefficients to arrive at the polynomial constraint database model. In this section, we study a property, called Property SL, which guarantees that n-dimensional semi-algebraic point-sets having Property SL are necessarily representable with linear polynomial equations and inequalities with *real* coefficients. We recall Definition 3.10 in which we defined an **R**-*semi-algebraic set* as an n-dimensional point-set representable with a polynomial constraint formula in which the polynomials are allowed to have real coefficients, and an **R**-*semi-linear set* as an

n-dimensional point-set representable with linear constraint formulae in which the linear constraints are allowed to have real coefficients.

We now formulate Property SL.

Definition 4.1 Let $S \subseteq \mathbf{R}^n$. We say that S has Property SL if, for every point $\vec{p}$ of $\overline{S}$, there exists a neighborhood V of $\vec{p}$ such that, for every point $\vec{q}$ of V,

1. if $\vec{q}$ is in S, then $]\vec{p}, \vec{q}[$ is fully contained within S; and
2. if $\vec{q}$ is not in S, then $]\vec{p}, \vec{q}[$ is disjoint from S. ■

We next claim the following.

Proposition 4.2 *Let S be an* **R***-semi-algebraic set. The set S is* **R***-semi-linear if and only if it has* Property SL*.*

The proof of this claim, given by Freddy Dumortier [20], is very technical, and therefore deferred to Appendix A, which can be read at the reader's discretion. The proof technique, which uses regular stratification of a semi-algebraic set, is of interest in its own right, and is the basis of some of the more algorithmic decomposition techniques of *semi-linear* sets later on in this chapter. Therefore, we will illustrate the decomposition technique of semi-linear sets used in the proof of Property SL by some examples. For the correctness of our technique, however, we refer to the proofs of the various lemmas in Appendix A.

Before doing so, however, we wish to give the reader a better understanding of Property SL.

First, we invite the reader to convince himself or herself that **R**-semi-linear sets have Property SL, e.g., by verifying this property for the **R**-semi-linear sets in Example 3.8, Figure 3.1, and in Example 3.16, Figure 3.4.

Second, we illustrate that non-**R**-semi-linear **R**-semi-algebraic sets do *not* have Property SL.

Example 4.3 Consider $S_1 = \{(x, y) \mid x \geq 0 \wedge y \geq 0 \wedge x^2 + y^2 = 4\}$ (Figure 4.1, *left*) and $S_2 = \{(x, y) \mid 0 \leq x \leq 3 \wedge 0 \leq y \leq 3 \wedge x^2 + y^2 \neq 4\}$ (Figure 4.1, *right*). The set S_1 is a quarter circle, and the set S_2 is a square with a quarter circle cut out. Both are non-**R**-semi-linear semi-algebraic sets. The set S_1 fails Property SL, since no open line segment connecting two different points of S_1 is contained within S_1. The set S_2 fails Property SL, since each open line segment connecting two different points of the cut-out quarter circle meets S_2. Observe that the cut-out quarter circle belongs to $\overline{S}_2$. ■

Figure 4.1: The non-semi-linear semi-algebraic sets of Example 4.3.

Third, and finally, we illustrate that a *non*-**R**-semi-algebraic (whence non-**R**-semi-linear) set can have Property SL.

Example 4.4 Consider $\mathbf{Z} \subseteq \mathbf{R}$, the set of all integer numbers. Since $\mathbf{Z}$ consists of an infinite number of connected components, $\mathbf{Z}$ cannot be **R**-semi-algebraic (see, e.g., [11]). Clearly, $\mathbf{Z}$ has Property SL. Hence, $\mathbf{Z}$ is an example of an *unbounded* non-**R**-semi-algebraic set having Property SL. ■

A *bounded* (possibly non-**R**-semi-algebraic) set having Property SL is necessarily **R**-semi-linear, however.

Proposition 4.5 *A* bounded *set having* Property SL *is* **R***-semi-linear.*

The proof of Proposition 4.5 can be found in Appendix A.

We now turn to the decomposition techniques of possibly unbounded semi-algebraic sets as used in the proof of Property SL in Appendix A.

Let S be an **R**-semi-linear set of $\mathbf{R}^n$. First, the regular stratification $S_{10}, \ldots, S_{m0}$ of S is obtained. Next, for each $i = 1, \ldots, m$, the regular stratification $S_{i1}, \ldots, S_{ik_i}$ of the boundary of S_{i0} within its affine support, is computed. Clearly, the set $\mathcal{C} = \{S_{ij} \mid 1 \leq i \leq m \wedge 0 \leq j \leq m_i\}$ is a finite cover of $\bigcup_{i=1}^{m} \overline{S}_{i0} = \overline{S}$, but not necessarily a partition of $\overline{S}$, since, unlike $S_{10}, \ldots, S_{m0}$, the boundaries of $S_{10}, \ldots, S_{m0}$ within their affine supports, need not be disjoint. The decomposition $\mathcal{C}$ will be called the *canonical decomposition of* $\overline{S}$ *relative to* S.

We illustrate the notion of "canonical decomposition" by an example.

Example 4.6 Consider the semi-linear set

$$\begin{aligned} S \;=\; & \{(x,y) \mid (-2 < x < 4 \wedge -2 < y < 4 \wedge x \neq 1 \wedge \\ & \neg(x < 1 \wedge y = -1) \wedge \neg(x > 1 \wedge y = 1)) \vee (x = -2 \wedge y = 1)\} \end{aligned}$$

Figure 4.2: The semi-linear set of Example 4.10.

in the plane, shown in Figure 4.2.

Clearly, $\overline{S}$ is the 6×6 square centered around the point $(1, 1)$.

The regular stratification of S consists of 2 layers. The first layer contains the 4 open rectangles, and the second and final layer consists of the point $(-2, 1)$. In total, the regular stratification of S consists of 5 strata. The canonical decomposition $\mathcal{C}$ consists of these 5 regular strata and the regular stratification of the boundary of each stratum of S with respect to its affine support. We thus see that $\mathcal{C}$ consists of 4 open rectangles, the 4×4 open edges delineating them, the 4×4 corner points, and the point $(-2, 1)$. Hence, in total, $\mathcal{C}$ constitutes of 29 different sets. ■

Now, let $S \subseteq \mathbf{R}^n$ be a semi-linear set, and let $\mathcal{C}$ be the canonical decomposition of $\overline{S}$ relative to S. Denote by T_C, $C \in \mathcal{C}$, the affine support of C. Let $H_{C1}, \ldots, H_{Ck_C}$ be $(n-1)$-dimensional hyperplanes of $\mathbf{R}^n$ such that their intersection equals T_C[1]. For each $C \in \mathcal{C}$ and $i = 1, \ldots k_C$, let $\mathcal{P}_{Ci}$ be the partition of $\mathbf{R}^n$ consisting of the hyperplane H_{Ci} and the two open half-spaces it delineates. Let $\mathcal{P}$ be the coarsest common refinement of $\mathcal{P}_{Ci}$, $C \in \mathcal{C}$ and $1 \leq i \leq k_C$. Clearly, each cell[2] of $\mathcal{P}$ can be obtained by choosing, for each $C \in \mathcal{C}$ and for each $i = 1, \ldots, k_C$, one cell of $\mathcal{P}_{Ci}$, and then taking the intersection of the chosen cells. Hence, $\mathcal{P}$ is a finite partition of $\mathbf{R}^n$ whose cells are open convex polyhedra. Then, the $\mathbf{R}$-semi-linear set S can be written as a finite union of cells of $\mathcal{P}$. If $\mathcal{H}$ is the collection of all hyperplanes considered above, $\mathcal{P}$ is called the partition of $\mathbf{R}^n$ *induced* by $\mathcal{H}$.

The claim above follows directly from an inspection of the proofs of Lemmas A.6 and Proposition A.7. For future reference, we state and prove this claim formally.

[1]We consider $\mathbf{R}^n$ as the intersection of zero $(n-1)$-dimensional hyperplanes.

[2]We use the term *cell* to refer to an element of a partition.

Proposition 4.7 *Let $S \subseteq \mathbf{R}^n$ be an $\mathbf{R}$-semi-linear set and let $\mathcal{C}$ be the canonical decomposition of $\overline{S}$ relative to S. Let $\mathcal{T}$ be the set of the affine supports of the members of $\mathcal{C}$. Let $\mathcal{H}$ be a finite set of $(n-1)$-dimensional hyperplanes such that each member of $\mathcal{T}$ is an intersection of members of $\mathcal{H}$. Then, each regular stratum of S (whence S) is a union of cells of the partition $\mathcal{P}$ of $\mathbf{R}^n$ induced by $\mathcal{H}$.*

Proof. Let C be a regular stratum of S. By Lemma A.4, C is open within its affine support. Furthermore, $\mathcal{C}$ contains the regular stratification of $\partial C = \overline{C} - C$, which, again by Lemma A.4, constitutes a partition of ∂C in sets which are open within their affine supports. Thus, C and $\mathcal{H}$ meet the conditions of Lemma A.6. A straightforward inspection of the proof of Lemma A.6 shows that C is a union of cells of $\mathcal{P}$. ∎

The following example illustrates Proposition 4.7.

Example 4.8 Consider the semi-linear set

$$\begin{aligned} S \;=\; & \{(x,y) \mid (3x - y > 2 \wedge y > 1 \wedge x + 2y < 10) \;\wedge \\ & \neg(x \leq 3 \wedge y \geq 2 \wedge x \geq y) \wedge \neg(1 \leq y \leq 3 \wedge x = 3)\} \end{aligned}$$

in the plane, shown in Figure 4.3, which consists of an open triangle, out of which a closed triangle and a closed line segment have been cut out. Each point of S is regular, whence S itself is the only regular stratum in the regular stratification of S.

Figure 4.3: The semi-linear set of Example 4.8.

The affine support T_0 of S is the entire plane. We now consider ∂S, the topological boundary of S (within T_0), shown in Figure 4.4.

Clearly, ∂S is the disjoint union of its regular stratification which consists of the 8 open line segments and the 7 points defining them, indicated in Figure 4.4. They constitute the sequence $S_0, \ldots, S_m$ (whence $m = 15$). The respective affine supports of these 15 sets, $T_0, \ldots, T_m$, are the lines supporting the 8 line segments and the 7 points, supporting themselves.

Figure 4.4: The set ∂S in Example 4.8.

In the two-dimensional plane, hyperplanes are lines. The 6 lines shown in Figure 4.5 suffice to describe each of the affine supports $T_0, T_1, \ldots, T_m$ as an intersection of these lines. Indeed, T_0, which is the entire plane, is the empty intersection of lines. Next, $T_1, \ldots, T_8$ are precisely these 6 lines. Finally, each of the 7 points, $T_9, \ldots, T_{15}$, is the intersection of two lines which support non-collinear line segments meeting in that point.

The 6 lines shown in Figure 4.5, together with the open half-planes they delineate, induce a partition of the entire plain, consisting of 12 points, 31 open line segments or half-lines, and 20 open regions. In Figure 4.5, these regions have been identified by numbers. By construction, all 63 members of the induced partition are semi-linear sets.

Figure 4.5: The lines describing the affine supports $T_0, T_1 \ldots, T_m$ of $S_0, S_1, \ldots, S_m$, respectively, described in Example 4.8. The numbers indicate the open regions in the induced partition.

Finally, S is the union of some members of the induced partition. Indeed, we see in this example that S is the union of the open regions 13, 14, 15, 17, 18, and 19, all of which are filled polygons, and the open intervals separating them, except of course

the open interval separating region 13 and 19. ■

We stress that, in general, it does *not* suffice to consider only the regular stratification of the boundary of the regular strata in the top layer of the regular stratification of S in order to be able to write S as a finite union of cells of the partition induced by hyperplanes defining the affine supports of the elements of a decomposition of S. We invite the reader to verify this on Example 4.6.

Unfortunately, it is unlikely that the canonical decomposition of a semi-linear set can be computed in FO + poly, since the definition of the regular stratification of a semi-linear set involves connectivity. Therefore, we propose an algorithm in the following section which is a variation on the technique resulting in a canonical decomposition that does not require the expressibility of connectivity. The connected components of the output of that algorithm constitute what we shall call the *algorithmic decomposition*. Given a semi-linear set S, we show that, as for the canonical decomposition, any collection of hyperplanes defining the affine supports of the elements of an algorithmic decomposition of $\overline{S}$ relative to S induces a partition of $\mathbf{R}^n$ for which S is a finite union of cells.

4.2 Algorithmic Decompositions of Semi-Linear Sets

Given an **R**-semi-linear *semi-algebraic* set, we propose the following algorithm, implementable in FO + linear-**Z**. Recall that $\mathtt{reg}(S, \vec{x})$ computes the set of regular points of S in which S has local dimension the overall dimension of S.

Algorithm 4.9
Input: An **R**-semi-linear semi-algebraic set S of $\mathbf{R}^n$.
Output: A finite sequence $S^1, \ldots, S^k$ of subsets of $\mathbf{R}^n$.
Method:

1. Initialize i to 0;
2. Increment i;
3. Let $S^i = \{\vec{x} \mid \mathtt{reg}(S, \vec{x})\}$;
4. If $S - S^i$ is not empty, repeat Steps 2–5 with S replaced by $S - S^i$;
5. If $\overline{S^i} - S^i$ is not empty, repeat Steps 2–5 with S replaced by $\overline{S^i} - S^i$. ■

In words, Algorithm 4.9 first computes the regular stratification of S; then Algorithm 4.9 is recursively applied to the union of the boundaries of the strata in each layer of this regular stratification (rather than the boundaries of the individual strata). Also, Algorithm 4.9 returns the layers of the various regular stratifications it performs (rather than the individual strata). In this way, Algorithm 4.9 does not involve taking connected components.

The set of all *connected* components of sets in the output of Algorithm 4.9 is called the *algorithmic decomposition* of $\overline{S}$ relative to S.

In the following two examples, we illustrate Algorithm 4.9, and contrast the notions of algorithmic decomposition and canonical decomposition.

Example 4.10 Recall Example 4.6, in which we obtained a canonical decomposition of the semi-linear set S shown in Figure 4.2.

If we apply Algorithm 4.9 to S, we first obtain the layers of the regular stratification of S. Hence, S^1 consists of the 4 open rectangles in Figure 4.2 and S^2 consists of the point $(-2, 1)$. Since $\overline{S^2} - S^2 = \emptyset$, we must recursively apply Algorithm 4.9 only to $\overline{S^1} - S^1$, the union of the boundaries of the 4 open rectangles. We first obtain the layers of the regular stratification of $\overline{S^1} - S^1$. Hence, S^3 consists of the 13 open line segments contained in this union, and S^4 consists of the in total 10 end points of these open line segments. Since $\overline{S^4} - S^4$ is empty, we must recursively apply Algorithm 4.9 only to $\overline{S^3} - S^3 = S^4$. Clearly, $S^5 = S^4$, and the algorithm terminates. Hence, the connected components of S^1, S^2, S^3, and $S^4 = S^5$, respectively, 28 sets in total, constitute the algorithmic decomposition of $\overline{S}$ relative to S. Notice that the canonical and the algorithmic decomposition do not coincide. ■

Example 4.11 Consider the semi-linear set

$$S = \{(x, y, z) \mid (y = 0 \wedge x \neq 1) \vee (x = 1 \wedge 2y + z = 3 \wedge z \geq 1)\}$$

in three-dimensional space, shown in Figure 4.6. The reader is invited to verify that the canonical decomposition $\mathcal{C}$ consists of the two open half-planes in which the plane $y = 0$ is divided, the line $y = 0 \wedge x = 1$, the open half-line $x = 1 \wedge 2y + z = 3 \wedge z > 1$, and the point $(1, 1, 1)$, 5 sets in total.

If we apply Algorithm 4.9 to S, we first obtain the layers of the regular stratification of S. Hence, S^1 consists of the two open half-planes in which the plane $y = 0$ is divided, S^2 is the open half-line $x = 1 \wedge 2y + z = 3 \wedge z > 1$, and S^3 consists of the single point $(1, 1, 1)$. Since $\overline{S^3} - S^3 = \emptyset$, we must recursively apply Algorithm 4.9 only to $\overline{S^1} - S^1$, the line $y = 0 \wedge x = 1$, and $\overline{S^2} - S^2$, the point $(1, 1, 1)$. Since both sets are affine subspaces of $\mathbf{R}^3$, we see that the first recursive call results in only one layer, $S^4 = \overline{S^1} - S^1$, and the second recursive call also results in only one layer,

Figure 4.6: The semi-linear set of Example 4.11.

$S^5 = \overline{S^2} - S^2 = S^3$, after which the algorithm terminates. Hence, the connected components of S^1, S^2, $S^3 = S^5$, and S^4, respectively, 5 sets in total, constitute the algorithmic decomposition of $\overline{S}$ relative to S. In this example, the algorithmic and the canonical decomposition coincide. ∎

It turns out that the analogue of Proposition 4.7 for algorithmic decompositions holds, which is our justification for proposing Algorithm 4.9 and considering the notion of "algorithmic decomposition." In order to prove this analogue, we need the following strengthening of Lemma A.6.

Lemma 4.12 *Let $S \subseteq \mathbf{R}^n$ be a finite disjoint union of sets $S_0, \ldots, S_k$, each of which is open within its affine support. Let $\bigcup_{i=0}^{k} \partial S_i$ ($\partial S_i = \overline{S_i} - S_i$) be a finite disjoint union of sets $S_{k+1}, \ldots, S_m$, each of which is open within its affine support. Let $T_0, \ldots, T_m$ be the affine supports of $S_0, \ldots, S_m$, respectively. Let $\mathcal{H}$ be a finite set of $(n-1)$-dimensional hyperplanes such that, for $i = 0, \ldots, m$, T_i is an intersection of members of $\mathcal{H}$. Then, for $i = 1, \ldots, k$, S_i is a union of cells of the partition of $\mathbf{R}^n$ induced by $\mathcal{H}$.*

Proof. A straightforward verification reveals that the arguments developed in the proof of Lemma A.6 apply literally to the situation described in Lemma 4.12. ∎

We are now ready to prove the analogue of Proposition 4.7 for algorithmic decompositions.

Proposition 4.13 *Let $S \subseteq \mathbf{R}^n$ be an $\mathbf{R}$-semi-linear semi-algebraic set and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Let $\mathcal{T}$ be the set of the affine*

supports of the members of $\mathcal{D}$. Let $\mathcal{H}$ be a finite set of $(n-1)$-dimensional hyperplanes such that each member of $\mathcal{T}$ is an intersection of members of $\mathcal{H}$. Then, each member of $\mathcal{D}$ (whence S) is a union of cells of the partition of $\mathbf{R}^n$ induced by $\mathcal{H}$.

Proof. Let L be the layer (set in the output of Algorithm 4.9) to which D belongs. The set L is a finite disjoint union of sets $D = D_1, \ldots, D_m$, each of which is open within its affine support. Since, for $i, j = 1, \ldots, m$, $i \neq j$, $\overline{D_i} \cap D_j = \emptyset$,

$$\overline{L} - L = \bigcup_{i=1}^{m} \bigcap_{j=1}^{m} (\overline{D_i} - D_j) = \bigcup_{i=1}^{m} \overline{D_i} - D_i = \bigcup_{i=1}^{m} \partial D_i.$$

Now, the algorithmic decomposition of $\overline{S}$ relative to S contains an algorithmic decomposition of $\overline{\overline{L} - L} = \overline{L} - L$ relative to $\overline{L} - L$, which in turn contains a regular stratification of $\overline{L} - L$. This regular stratification of $\overline{L} - L$ is a partition of $\overline{L} - L$ in members of $\mathcal{D}$, each of which is open within its affine support. Thus, L and $\mathcal{H}$ meet the conditions of Lemma 4.12, whence, in particular, D is a union of cells of $\mathcal{P}$. ∎

We conclude this section with some properties of algorithmic decompositions which will prove useful in the following section.

Proposition 4.14 *Let $S \subseteq \mathbf{R}^n$ be an $\mathbf{R}$-semi-linear semi-algebraic set, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Let $D \in \mathcal{D}$. Suppose that $\partial\overline{D}$ (the boundary of $\overline{D}$ within its affine support) is not empty. Let R be a stratum in the top layer of the regular stratification of $\partial\overline{D}$.*

1. *There exists $D' \in \mathcal{D}$ with $D' \subseteq R$ and $\mathtt{dim}(D') = \mathtt{dim}(D) - 1$.*
2. *There exist pairwise disjoint members $D'_1, \ldots, D'_r$ of $\mathcal{D}$ in the same layer (a set in the output of Algorithm 4.9 applied to S) with, for $i = 1, \ldots, r$, $D'_i \subseteq R$ and $\mathtt{dim}(D'_i) = \mathtt{dim}(D) - 1$, such that $\overline{D'_1} \cup \ldots \cup \overline{D'_r} = \overline{R}$.*
3. *There exist pairwise disjoint members $D'_1, \ldots, D'_m$ in the same layer of $\mathcal{D}$ with, for $i = 1, \ldots, m$, $D'_i \subseteq \partial\overline{D}$ and $\mathtt{dim}(D'_i) = \mathtt{dim}(D) - 1$, such that $\overline{D'_1} \cup \ldots \cup \overline{D'_m} = \partial\overline{D}$.*

Proof. Let L be the layer to which D belongs. Since, clearly, $\overline{D} \cap (L - D) = \emptyset$, $\partial D = \overline{D} - D \subseteq \overline{L} - L$. Hence, ∂D, whence $\partial\overline{D}$, whence R is covered by the members of $\mathcal{D}$ that belong to the regular stratification of $\overline{L} - L$. All these members are of strictly lower dimension than D. Let $\mathtt{dim}(D) = d$. Clearly, $\mathtt{dim}(R) = \mathtt{dim}(\partial\overline{D}) = d - 1$. We are now ready to prove the three claims.

1. Since R is covered by a finite number of members of $\mathcal{D}$ of dimension at most $d-1$, at least one of them, say D', satisfies $\mathtt{dim}(R \cap D') = d-1$. In particular, $\mathtt{dim}(D') = d-1$ and R and D' have the same affine support, say T. Suppose D' contains points outside R. Since D' is connected, D' must contain a point on the boundary of R within T. However, any neighborhood of such a point must necessarily contain points of $\partial\overline{D}$ outside T, whence, $\partial\overline{D}$ does not look like a $(d-1)$-dimensional affine subspace in any neighborhood of such a point, a contradiction. Hence, $D' \subseteq R$.

2. Let $D'_1, \ldots, D'_r$ be the strata in the top layer of the regular stratification of $\overline{L} - L$ (these are members of $\mathcal{D}$) for which, for $i = 1, \ldots, r$, $\mathtt{dim}(R \cap D'_i) = d-1$. By the arguments used in the proof of the first claim, $D'_i \subseteq R$. Since R is covered by the regular stratification of $\overline{L} - L$, $\mathtt{dim}(R - (D'_1 \cup \ldots \cup D'_r)) < d-1$, from which the second claim immediately follows.

3. For the third claim, we first observe that the union of the strata in the top layer of the regular stratification of $\partial\overline{D}$ is dense in $\partial\overline{D}$. Second, we observe that the layer to which $D'_1, \ldots, D'_r$ in the second claim belong does not depend on the choice of R. The third claim now immediately follows. ∎

Proposition 4.15 *Let S_1, S_2, and B be $\mathbf{R}$-semi-linear semi-algebraic sets of $\mathbf{R}^n$ such that B is open within $\mathbf{R}^n$ and $S_1 \cap B = S_2 \cap B$. For each set S_1^i in the output of Algorithm 4.9 applied to S_1 such that $S_1^i \cap B$ is not empty, there exists a set S_2^j in the output of Algorithm 4.9 applied to S_2 such that $S_2^j \cap B$ is not empty satisfying $S_1^i \cap B = S_2^j \cap B$, and vice-versa.*

Proof. We prove by induction on the dimension of $d = \mathtt{dim}(S_1)$ that, for each set S_1^i in the output of Algorithm 4.9 applied to S_1 such that $S_1^i \cap B$ is not empty, there exists a set S_2^j in the output of Algorithm 4.9 applied to S_2 such that $S_2^j \cap B$ is not empty satisfying $S_1^i \cap B = S_2^j \cap B$. Proposition 4.15 then follows because of symmetry reasons.

Suppose $d = 0$. Then S_1 consists of isolated points only, whence also $S_1 \cap B = S_2 \cap B$ consists of isolated points only. Clearly, S_1 is the only set in the output of Algorithm 4.9 applied to S_1. The zero-dimensional layer of the regular stratification of S_2, which occurs in the output of Algorithm 4.9 applied to S_2, necessarily contains all points of $S_1 \cap B = S_2 \cap B$. Hence, Proposition 4.15 is satisfied.

Suppose $d > 0$. First, we observe that each set in the output of Algorithm 4.9 applied to S_1 (respectively, S_2) with dimension higher than $\mathtt{dim}(S_1 \cap B) = \mathtt{dim}(S_2 \cap B)$ has an empty intersection with B. We now distinguish two cases.

1. $\mathtt{reg}(S_1) \cap B = \emptyset$. Then, for $\mathtt{reg}(S_1)$, there are no further conditions to be satisfied. Next, Algorithm 4.9 has to be applied recursively to $S_1 - \mathtt{reg}(S_1)$, unless that set is empty. Clearly, since $(S_1 - \mathtt{reg}(S_1)) \cap B = S_1 \cap B = S_2 \cap B$ and $\mathtt{dim}(S_1 - \mathtt{reg}(S_1)) < d$, $S_1 - \mathtt{reg}(S_1)$, S_2, and B satisfy Proposition 4.15 by the induction hypothesis. Finally, Algorithm 4.9 has to be applied recursively to $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$, unless that set is empty. However, $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$ has an empty intersection with B, since $\mathtt{reg}(S_1) \cap B = \emptyset$ implies that also $\overline{\mathtt{reg}(S_1)} \cap B = \emptyset$. Hence, there are no further conditions to be satisfied.

2. $\mathtt{reg}(S_1) \cap B \neq \emptyset$. Let $d = \mathtt{dim}(S_1 \cap B) = \mathtt{dim}(S_2 \cap B)$. Let S_2' be the set S_2 of which all layers of the regular stratification of S_2 of dimension strictly higher than d are stripped off. Notice that the algorithmic decomposition of $\overline{S_2'}$ relative to S_2' is contained in the algorithmic decomposition of $\overline{S_2}$ relative to S_2. Clearly, $S_1 \cap B = S_2 \cap B = S_2' \cap B$, and $\mathtt{dim}(S_1) = \mathtt{dim}(S_1 \cap B) = \mathtt{dim}(S_2' \cap B) = \mathtt{dim}(S_2') = d$. By using neighborhoods fully contained within the open set B, it is readily seen that $\mathtt{reg}(S_1) \cap B = \mathtt{reg}(S_1 \cap B) = \mathtt{reg}(S_2' \cap B) = \mathtt{reg}(S_2') \cap B$. The two layers under consideration satisfy the required correspondence.

   Next, Algorithm 4.9 has to be applied recursively to $S_1 - \mathtt{reg}(S_1)$, unless that set is empty (respectively, $S_2' - \mathtt{reg}(S_2')$, unless that set is empty). We first observe that $(S_1 - \mathtt{reg}(S_1)) \cap B = (S_1 \cap B) - (\mathtt{reg}(S_1) \cap B) = (S_2' \cap B) - (\mathtt{reg}(S_2') \cap B) = (S_2' - \mathtt{reg}(S_2')) \cap B$. Since this latter set is empty if either $S_1 - \mathtt{reg}(S_1)$ or $S_2' - \mathtt{reg}(S_2')$ are empty, there are no further conditions to be satisfied in this case. If neither $S_1 - \mathtt{reg}(S_1)$ nor $S_2' - \mathtt{reg}(S_2')$ are empty, then $\mathtt{dim}(S_1 - \mathtt{reg}(S_1)) < d$, and $S_1 - \mathtt{reg}(S_1)$, $S_2' - \mathtt{reg}(S_2')$, and B satisfy Proposition 4.15 by the induction hypothesis.

   Finally, Algorithm 4.9 has to be applied recursively to $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$, unless that set is empty (respectively, $\overline{\mathtt{reg}(S_2')} - \mathtt{reg}(S_2')$, unless that set is empty). If A is an arbitrary set and G an open set of an arbitrary topological space, the identity $\overline{A} \cap G = \overline{A \cap G} \cap G$ holds. Using this identity, we observe that

$$\begin{aligned}
(\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)) \cap B &= (\overline{\mathtt{reg}(S_1)} \cap B) - (\mathtt{reg}(S_1) \cap B) \\
&= (\overline{\mathtt{reg}(S_1) \cap B} \cap B) - (\mathtt{reg}(S_1) \cap B) \\
&= (\overline{\mathtt{reg}(S_2') \cap B} \cap B) - (\mathtt{reg}(S_2') \cap B) \\
&= (\overline{\mathtt{reg}(S_2')} \cap B) - (\mathtt{reg}(S_2') \cap B) \\
&= (\overline{\mathtt{reg}(S_2')} - \mathtt{reg}(S_2')) \cap B.
\end{aligned}$$

   Since this set is empty if either $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$ or $\overline{\mathtt{reg}(S_2')} - \mathtt{reg}(S_2')$ are empty, there are no further conditions to be satisfied in this case. If neither $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$ nor $\overline{\mathtt{reg}(S_2')} - \mathtt{reg}(S_2')$ are empty, then $\mathtt{dim}(\overline{\mathtt{reg}(S_1)} -$

$\mathtt{reg}(S_1)) < d$, and $\overline{\mathtt{reg}(S_1)} - \mathtt{reg}(S_1)$, $\overline{\mathtt{reg}(S_2')} - \mathtt{reg}(S_2')$, and B satisfy Proposition 4.15 by the induction hypothesis. ∎

A special case of Proposition 4.15 is the following.

Proposition 4.16 *Let S and B be $\mathbf{R}$-semi-linear semi-algebraic sets of $\mathbf{R}^n$ such that B is open within $\mathbf{R}^n$. For each set S^i in the output of Algorithm 4.9 applied to S such that $S^i \cap B$ is not empty, there exists a set S_B^j in the output of Algorithm 4.9 applied to $S \cap B$ such that $S_B^j \cap B$ is not empty satisfying $S^i \cap B = S_B^j \cap B$, and vice-versa.*

Proof. The sets S, $S \cap B$, and B satisfy Proposition 4.15. ∎

4.3 A-Semi-Linearity of Semi-Algebraic Sets is Decidable

We intend to use the notion of algorithmic decomposition, in particular Proposition 4.13 and the fact that Algorithm 4.9 can be implemented in FO + linear-$\mathbf{Z}$ (whence certainly in FO + poly) to show that $\mathbf{R}$-semi-linear semi-algebraic sets are $\mathbf{A}$-semi-linear, as a consequence of which Property SL (which can easily be expressed in FO + poly) decides whether a semi-algebraic set is $\mathbf{A}$-semi-linear.

To prove this, we first consider *bounded* semi-algebraic sets.

Lemma 4.17 *Let S be a bounded $\mathbf{R}$-semi-linear semi-algebraic set of $\mathbf{R}^n$, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Then all the affine supports of the members of $\mathcal{D}$ are $\mathbf{A}$-semi-linear.*

Proof. Let S^0 be the union of the zero-dimensional sets in the output of the algorithm, which is also the union of all members of $\mathcal{D}$ consisting of a single point. We shall call these points *special points*. Since Algorithm 4.9 can be implemented in FO + linear-$\mathbf{Z}$ (whence in FO + poly), $S^1, \ldots, S^k$ are semi-algebraic. Since it can be decided in FO + linear-$\mathbf{Z}$ whether a semi-algebraic set is zero-dimensional, i.e., consists of isolated points only (see Section 3.1), S^0 is semi-algebraic. Clearly, all members of $\mathcal{D}$ consisting of a single point are the intersection of S^0 and a suitable semi-algebraic neighborhood of that point, and must therefore be semi-algebraic, too. Thus, each special point can be described by a real constraint formula. By

the Tarski-Seidenberg quantifier elimination theorem [69, 71], it follows that the coordinates of that point can be described using univariate polynomials with integer coefficients. By definition, it follows that these coordinates must be real algebraic numbers.

Now, let D be a member of $\mathcal{D}$, and let T be the affine support of D. We claim that T is the affine support of a set of special points. Since these special points have real algebraic coordinates, it then follows that the standard techniques from analytical geometry to obtain a system of linear equations describing their affine support T only generate real algebraic coefficients. Consequently, T is **A**-semi-linear, what has to be shown.

We prove the above claim by induction on the dimension of D.

For the basis of this induction, we observe that, if D is zero-dimensional, the claim becomes trivial.

Now assume that the claim holds for all members of $\mathcal{D}$ of dimension strictly less than d, $0 < d \leq \mathtt{dim}(S)$.

Let D be a d-dimensional member of $\mathcal{D}$, and let T be the affine support of D. Clearly, T is also the affine support of $\overline{D}$. Let $D_1, \ldots, D_m$ be the members of $\mathcal{D}$ fully contained within $\partial\overline{D}$ (the boundary of $\overline{D}$ within T, which is not empty since D, whence $\overline{D}$, is bounded), and let $T_1, \ldots, T_m$ be their respective affine supports. By the third claim of Proposition 4.14, $\overline{D_1} \cup \ldots \cup \overline{D_m} = \partial\overline{D}$. Since $\overline{D}$ is bounded and the closure of a set open within T, it follows that T is also the affine support of $\partial\overline{D}$, whence also of $\overline{D_1} \cup \ldots \cup \overline{D_m}$, whence also of $D_1 \cup \ldots \cup D_m$. By the induction hypothesis, we know that $T_1, \ldots, T_m$ are the respective affine supports of some subsets $S_1^0, \ldots, S_m^0$ of S^0. Hence, T is the affine support of $S_1^0 \cup \ldots \cup S_m^0$, also a subset of S^0. ∎

From the proof of Lemma 4.17, we obtain an algorithm, implementable in FO + **Z**-linear, which outputs the set of *special points* of a *bounded* **R**-semi-linear semi-algebraic set S.

Algorithm 4.18
Input: A *bounded* **R**-semi-linear semi-algebraic set S of $\mathbf{R}^n$.
Output: The finite set S^0 of *special points* of S.
Method:

1. Initialize S^0 to the empty set;
2. Let $S^r = \{\vec{x} \mid \mathtt{reg}(S, \vec{x})\}$;
3. If S^r is zero-dimensional, then let $S^0 = S^0 \cup S^r$;

4. If $S - S^r$ is not empty, repeat Steps 2–5 with S replaced by $S - S^r$;

5. If $\overline{S^r} - S^r$ is not empty, repeat Steps 2–5 with S replaced by $\overline{S^r} - S^r$. ■

A closer inspection of the proof of Lemma 4.17, in combination with Proposition 4.13, immediately reveals the following result.

Proposition 4.19 *Let S be a* bounded $\mathbf{R}$*-semi-linear semi-algebraic set of* $\mathbf{R}^n$*. Let S^0 be the finite set of special points of S returned by Algorithm 4.18. Let $\mathcal{H}$ be a finite set of $(n-1)$-dimensional hyperplanes such that every affine support of a subset of points of S_0 can be described as an intersection of members of $\mathcal{H}$. Then, S is a union of cells of the partition of $\mathbf{R}^n$ induced by $\mathcal{H}$.*

We next wish to prove that Lemma 4.17 also holds for *unbounded* $\mathbf{R}$-semi-linear semi-algebraic sets. In order to prove this generalization, we introduce the notion of *bounding box*.

Definition 4.20 Let S be an $\mathbf{R}$-semi-linear semi-algebraic set of $\mathbf{R}^n$. Let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. A *bounding box* for S is an open n-dimensional box B which intersects each member of $\mathcal{D}$. ■

To motivate our notion of bounding box, we first observe the following.

Lemma 4.21 *Let S be an $\mathbf{R}$-semi-linear semi-algebraic set, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Each* bounded *member of $\mathcal{D}$ is fully contained within any bounding box for S.*

Proof. Let B be any bounding box for S, and let D be a bounded member of $\mathcal{D}$. We prove Lemma 4.21 by induction on $\mathtt{dim}(D)$.

Let $\mathtt{dim}(D) = 0$. Then, D is a point, which, by definition, is contained in B.

Let $\mathtt{dim}(D) = d > 0$. Let $D_1, \ldots, D_m$ be the members of $\mathcal{D}$ contained in $\partial\overline{D}$, the boundary of $\overline{D}$ within its affine support, which is not empty since D, whence $\overline{D}$, is bounded. By the third claim of Proposition 4.14, $\overline{D_1} \cup \ldots \cup \overline{D_m} = \partial\overline{D}$. Since, for $i = 1, \ldots, m$, $D_i \subseteq B$, by the induction hypothesis, $\partial\overline{D} \subseteq \overline{B}$. Since $\overline{D}$ is bounded, it follows that $\overline{D} \subseteq \overline{B}$. Since $D \cap B \neq \emptyset$ and D is open within its affine support, D cannot contain points on the boundary of B. Hence $D \subseteq B$. ■

An immediate corollary to Lemma 4.21, which one might intuitively have desired, is the following.

Proposition 4.22 *Let S be a* bounded **R**-*semi-linear semi-algebraic set. Then S is fully contained within any bounding box for S.*

We now present three examples illustrating Definition 4.20 for *unbounded* semi-linear sets.

Example 4.23 Consider the unbounded semi-linear S of Example 4.11, shown in Figure 4.6, and the algorithmic decomposition $\mathcal{D}$ of $\overline{S}$ relative to S that was obtained. The open $4 \times 4 \times 4$ cube B centered around $\vec{0}$ intersects all five members of $\mathcal{D}$, and is therefore a bounding box for S. ■

Example 4.24 Consider the semi-linear set $S = \{(x, y) \mid x \geq 0 \wedge 0 < 2y \leq x\}$ in the plane, shown as the heavily shaded angular sector in Figure 4.7.

Figure 4.7: The semi-linear set of Example 4.24.

The reader is invited to verify that the algorithmic decomposition $\mathcal{D}$ of $\overline{S}$ relative to S consists of the open angular sector, the two open half-lines in the boundary of the angular sector, and the angular point.

Now, let B be the open 2×2 square centered around $\vec{0}$, shown in lighter shading in Figure 4.7. Since B intersects all four members of $\mathcal{D}$, B is a bounding box for S. ■

Example 4.25 Consider the semi-linear set $S = \{(x, y) \mid -1 \leq x - y \leq 1\}$ in the plane, shown as the heavily shaded strip in Figure 4.8.

The reader is invited to verify that the algorithmic decomposition $\mathcal{D}$ of $\overline{S}$ relative to S consists of the open strip and the two lines delineating it, none of which is bounded.

Now, let B again be the open 2×2 square centered around $\vec{0}$, shown in lighter shading in Figure 4.7. Since B intersects all three members of $\mathcal{D}$, B is a bounding box for S. ■

Figure 4.8: The semi-linear set of Example 4.25.

Given an **R**-semi-linear semi-algebraic set S of $\mathbf{R}^n$, we now provide an algorithm, implementable in FO + **Z**-linear, which computes a bounding box for S.

For that purpose, we consider all the 2^n different open **Z**-semi-linear sets $\sigma_1 x_1 < 1 \wedge \ldots \wedge \sigma_n x_n < 1$, with, for $i = 1, \ldots, n$, $\sigma_i = -1$ or $\sigma_i = +1$. If $\vec{p} = (\sigma_1, \ldots, \sigma_n)$, then this set is the translation of the open coordinate hyperquadrants $\sigma_1 x_1 < 0 \wedge \ldots \wedge \sigma_n x_n < 0$ over the vector $\vec{p}$. As a consequence of these translations, the sets obtained cover $\mathbf{R}^n$. We shall denote these sets by $E_1, \ldots, E_{2^n}$, and still use the term *hyperquadrant* (or *quadrant*, if $n = 2$) to refer to them.

Example 4.26 Figure 4.9 shows the quadrants E_1, E_2, E_3, and E_4 of $\mathbf{R}^2$. ■

Figure 4.9: The quadrants E_1, E_2, E_3, and E_4 of $\mathbf{R}^n$.

Algorithm 4.27
Input: An **R**-semi-linear semi-algebraic set S of $\mathbf{R}^n$.
Output: A bounding box for S.
Method:

1. Let $E_1, \ldots, E_{2^n}$ be the hyperquadrants of $\mathbf{R}^n$ defined above.
2. Initialize S_E^0 to the empty set;
3. For $j = 1 \ldots 2^n$ do
   (a) $S_j = S \cap E_j$;
   (b) $S_j^r = \{\vec{x} \mid \mathtt{reg}(S_j, \vec{x})\}$;
   (c) If S_j^r is zero-dimensional, $S_E^0 = S_E^0 \cup S_j^r$;
   (d) If $S_j - S_j^r$ is not empty, repeat Steps 3b–3e with S_j replaced by $S_j - S_j^r$;
   (e) If $\overline{S_j^r} - S_j^r$ is not empty, repeat Steps 3b–3e with S replaced by $\overline{S_j^r} - S_j^r$;
4. Let r be the maximum of the absolute values of the coordinates of the points of S_E^0;
5. Let B be the $2(r+1) \times \ldots \times 2(r+1)$ n-dimensional hypercube centered around $\vec{0}$. ■

Notice that Algorithm 4.27 requires the computation of the algorithmic decompositions (Algorithm 4.9) of $\overline{S \cap E_j}$ relative to $S \cap E_j$, $1 \leq j \leq n$. In particular, S_E^0 is the union of the zero-dimensional members of these algorithmic decompositions.

Before proving the correctness of Algorithm 4.27, we illustrate it with three examples.

Example 4.28 Consider again the semi-linear set S of Example 4.11, shown in Figure 4.6. Let E_1 be the hyperquadrant $-x > 1 \wedge -y > 1 \wedge -z > 1$. We invite the reader to verify that the zero-dimensional members of the algorithmic decomposition of $\overline{S \cap E_1}$ relative to $S \cap E_1$ consists of the points $(-1, 0, -1)$, $(1, 0, -1)$, $(1, 1, 1)$, and $(1, -1, 6)$. We shall not explicitly consider the 7 other hyperquadrants $E_2, \ldots, E_7$; however, we may conclude that $r \geq 6$, and that the box B returned by Algorithm 4.27 containts the open $14 \times 14 \times 14$ cube B centered around $\vec{0}$. Hence, B is a superset of the bounding box found in Example 4.23, and, therefore, itself a bounding box for S. ■

Example 4.29 Consider again the semi-linear set S of Example 4.24, shown in Figure 4.7. Let the quadrants $E_1, \ldots, E_4$ be as shown in Figure 4.9. Clearly, $S \cap E_1 = S$. In Example 4.24, we found that the algorithmic decomposition $\mathcal{D}$ of $\overline{S}$ relative to S consists of the open angular sector, the two open half-lines in the boundary of the angular sector, and the angular point. Hence, the point $(0, 0)$ constitutes the only zero-dimensional set in the output of Algorithm 4.9 applied to $S \cap E_1$. We

invite the reader to verify that the points $(0,0)$, $(1,0)$, and $(1,0.5)$ constitute the zero-dimensional sets in the output of Algorithm 4.9 applied to both $S \cap E_2$ and $S \cap E_3$, and that the points $(0,0)$, and $(2,1)$ constitute the zero-dimensional sets in the output of Algorithm 4.9 applied to $S \cap E_4$. Hence, $r = 2$, and Algorithm 4.27 returns the 6×6 square B centered around $\vec{0}$, which is a superset of the bounding box of S shown in Figure 4.7, and, therefore, itself a bounding box. ■

Example 4.30 Consider again the semi-linear set S of Example 4.25, shown in Figure 4.8. Let the quadrants $E_1, \ldots, E_4$ be as shown in Figure 4.9. The reader is invited to verify that the zero-dimensional members of the algorithmic decomposition of $\overline{S \cap E_1}$ relative to $S \cap E_1$ consist of the points $(0,-1)$, $(-1,-1)$, and $(-1,0)$; that the zero-dimensional members of the algorithmic decomposition of $\overline{S \cap E_2}$ relative to $S \cap E_2$ consist of the points $(-2,-1)$, $(0,-1)$, $(1,0)$, and $(1,2)$; that the zero-dimensional members of the algorithmic decomposition of $\overline{S \cap E_3}$ relative to $S \cap E_3$ consist of the points $(1,0)$, $(1,1)$, and $(0,1)$; and that the zero-dimensional members of the algorithmic decomposition of $\overline{S \cap E_4}$ relative to $S \cap E_4$ consist of the points $(2,1)$, $(0,1)$, $(-1,0)$, and $(-1,-2)$. Hence, $r = 2$, and Algorithm 4.27 returns the 6×6 square B centered around $\vec{0}$, which is a superset of the bounding box of S shown in Figure 4.7, and, therefore, itself a bounding box. ■

We now prove the correctness of Algorithm 4.27.

Proposition 4.31 *Let S be an arbitrary* $\mathbf{R}$*-semi-linear semi-algebraic set of* $\mathbf{R}^n$*. Algorithm 4.27 is implementable in* FO + $\mathbf{Z}$-linear *and, upon input S, computes a bounding box for S.*

Proof. We first argue that Algorithm 4.27 is implementable in FO + $\mathbf{Z}$-linear. The sets $E_1, \ldots, E_{2^n}$ are described by $\mathbf{Z}$-linear constraint formulae. Since Algorithm 4.9 can be implemented in FO + linear-$\mathbf{Z}$, and since it can be checked in FO + $\mathbf{Z}$-linear whether a set is zero-dimensional, (see Section 3.1), it follows that S_E^0 can be computed from S in FO + linear-$\mathbf{Z}$. Clearly, the open n-dimensional hypercube B can then be computed from S_E^0 in FO + linear-$\mathbf{Z}$.

Second, we show that B is a bounding box for each $S \cap E_j$, $1 \leq j \leq 2^n$.

Thereto, let $\mathcal{D}_j$ be the algorithmic decomposition of $\overline{S \cap E_j}$ relative to $S \cap E_j$. To prove our claim, we must show that B intersects each member D of $\mathcal{D}_j$. We prove this by induction on $d = \mathtt{dim}(D)$.

Let $d = 0$. Then, by an earlier remark, D is contained within S_E^0, whence $D \subseteq B$.

Let $d > 0$. Since $\overline{D} \subseteq \overline{E_j}$, $\overline{D}$ cannot be an affine subspace of $\mathbf{R}^n$. Hence, $\partial\overline{D}$, the boundary of $\overline{D}$ within $\mathbf{R}^n$, is not empty. By Proposition 4.14, there exists a member

D' of $\mathcal{D}_j$ such that $D' \subseteq \partial\overline{D}$ and $\mathtt{dim}(D') = d - 1$. By the induction hypothesis, $D' \cap B \neq \emptyset$. Since B is open, it follows that $D \cap B \neq \emptyset$.

Third and finally, we show that B is a bounding box for S.

Let $\mathcal{D}$ the algorithmic decomposition of $\overline{S}$ relative to S. We show that B intersects each member D of $\mathcal{D}$. Let L be the layer (a set in the output of Algorithm 4.9 applied to S) to which D belongs. Since the hyperquadrants $E_1, \ldots, E_{2^n}$ cover $\mathbf{R}^n$, there exists j, $1 \leq j \leq 2^n$, such that $D \cap E_j \neq \emptyset$. Let $\mathcal{D}_j$ the algorithmic decomposition of $\overline{S \cap E_j}$ relative to $S \cap E_j$. Since E_j is open within $\mathbf{R}^n$, Proposition 4.16 applies. Hence, there exists a layer L_j (a set in the output of Algorithm 4.9 applied to $S \cap E_j$) such that $L \cap E_j = L_j \cap E_j$. Since L and L_j both consist of connected sets of the same dimension which are open within their respective affine supports, there exists a member D_j of $\mathcal{D}$ fully contained within $L_j \cap E_j$ such that $D_j \subseteq D$. Since B is a bounding box for $S \cap E_j$, $D_j \cap B \neq \emptyset$, whence $D \cap B \neq \emptyset$. ∎

We are now ready to generalize Lemma 4.17.

Lemma 4.32 *Let S be a $\mathbf{R}$-semi-linear semi-algebraic set of $\mathbf{R}^n$, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Then all the affine supports of the members of $\mathcal{D}$ are $\mathbf{A}$-semi-linear.*

Proof. If S is bounded, Lemma 4.32 follows from Lemma 4.17.

Let B be a bounding box for S computed from S in FO+linear-$\mathbf{Z}$ (Proposition 4.31). Clearly, $S \cap B$ is $\mathbf{R}$-semi-linear, as it is the intersection of two $\mathbf{R}$-semi-linear sets. Moreover, $S \cap B$ is semi-algebraic, as it can be computed from S in FO + linear-$\mathbf{Z}$ (whence in FO + poly).

Let $\mathcal{D}_B$ be the algorithmic decomposition of $\overline{S \cap B}$ relative to $S \cap B$. By Lemma 4.17, all the affine supports of the members of $\mathcal{D}_B$ are $\mathbf{A}$-semi-linear.

Now, let D be a member of $\mathcal{D}$, and let T be the affine support of D. Let L be the layer (a set in the output of Algorithm 4.9 applied to S) to which D belongs. Since $D \cap B \neq \emptyset$, $L \cap B \neq \emptyset$, whence, by Proposition 4.16, there exists a layer L_B (set in the output of Algorithm 4.9 applied to $S \cap B$) such that $L \cap B = L_B \cap B$. As in the final part of the proof of Proposition 4.31, we can find a member D_B of $\mathcal{D}_B$ fully contained within $L_B \cap B$ such that $D_B \subseteq D$. Since $\mathtt{dim}(D_B) = \mathtt{dim}(L_B) = \mathtt{dim}(L) = \mathtt{dim}(D)$, it follows that T is also the affine support of D_B. Since D_B is a member of $\mathcal{D}_B$, T is $\mathbf{A}$-semi-linear. ∎

We are now able to generalize Algorithm 4.18 for *arbitrary* $\mathbf{R}$-semi-linear semi-algebraic sets.

Algorithm 4.33
Input: An **R**-semi-linear semi-algebraic set S of $\mathbf{R}^n$.
Output: The finite set S^0 of *special points* of S.
Method:

1. Let B be the bounding box for S computed by Algorithm 4.27;
2. Let $S_B = S \cap B$;
3. Initialize S_B^0 to the empty set;
4. Let $S_B^r = \{\vec{x} \mid \mathtt{reg}(S_B, \vec{x})\}$;
5. If S_B^r is zero-dimensional, then let $S_B^0 = S_B^0 \cup S_B^r$;
6. If $S_B - S_B^r$ is not empty, repeat Steps 4–7 with S_B replaced by $S_B - S_B^r$;
7. If $\overline{S_B^r} - S_B^r$ is not empty, repeat Steps 4–7 with S_B replaced by $\overline{S_B^r} - S_B^r$. ■

We call S_B^0 the set of *special points* of S. Notice that this definition is consistent with our earlier use of the term "special points" in the context of bounded **R**-semi-linear semi-algebraic sets, as these are contained in any of their bounding boxes (Proposition 4.22). Hence, Algorithms 4.18 and 4.33 return the same set of points in this case.

A closer inspection of the proof of Lemma 4.32, in combination with Propositions 4.13 and 4.19, immediately reveals the following result.

Proposition 4.34 *Let S be an* arbitrary **R**-*semi-linear semi-algebraic set of $\mathbf{R}^n$. Let S_B^0 be the finite set of special points of S returned by Algorithm 4.33. Let $\mathcal{H}$ be a finite set of $(n-1)$-dimensional hyperplanes such that every affine support of a subset of points of S_B^0 can be described as an intersection of members of $\mathcal{H}$. Then, S is a union of cells of the partition of $\mathbf{R}^n$ induced by $\mathcal{H}$.*

Lemma 4.32 can be sharpened to Proposition 4.35, below.

Proposition 4.35 *Let S be an* **R**-*semi-linear semi-algebraic set of $\mathbf{R}^n$, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Then all members of $\mathcal{D}$ are* **A**-*semi-linear.*

Proof. Let $\mathcal{T}$ be the set of all the affine supports of the members of $\mathcal{D}$. By Lemma 4.32, all members of $\mathcal{T}$ are **A**-semi-linear. Hence, there exists a finite set $\mathcal{H}$ of $(n-1)$-dimensional **A**-semi-linear hyperplanes such that each member of $\mathcal{T}$ is an intersection of members of $\mathcal{H}$. Consider the finite partition $\mathcal{P}$ of $\mathbf{R}^n$ induced by $\mathcal{H}$. Obviously, all cells of $\mathcal{P}$ are **A**-semi-linear. By Proposition 4.13, each member of $\mathcal{D}$ is a union of cells of $\mathcal{P}$, and is therefore **A**-semi-linear, too. ■

Corollary 4.36 *Every* **R***-semi-linear semi-algebraic set is* **A***-semi-linear.*

We are now ready to state and prove our first main result.

Theorem 4.37 *Let S be a semi-algebraic set. The set S is* **A***-semi-linear if and only if it has* Property SL. *Moreover,* **A***-semi-linearity of a semi-algebraic set defined by a polynomial constraint formula is decidable by an* FO + poly *formula.*

Proof. The first statement in Theorem 4.37 is an immediate consequence of Proposition 4.2 and Corollary 4.36. The second statement follows from the first, because the corresponding Boolean decision query of type $[0, n] \to [0, 0]$ can easily be expressed in FO + poly using Property SL, and the validity of polynomial constraint sentences in **R** is decidable. ■

It is important to note that the truth of the first statement in Theorem 4.37 is *not* revealed by the proof of Proposition 4.2, because Definition 3.12 of regular point does not specify anything regarding the type of the coefficients in the polynomials involved. To obtain this information, we made a more thorough study above of the decomposition process outlined in the proof of Proposition 4.2.

4.4 Z-Semi-Linearity of Semi-Algebraic Sets is Decidable

We now put to use the results of Section 4.2 a second time to show that **Z**-semi-linearity of a semi-algebraic set is decidable, too. Then, we show that the corresponding decision query can*not* be expressed in FO + poly, however.

First, we summarize the results we obtain if we apply the same line of argument as in Section 4.3 to **Z**-semi-linearity instead of **A**-semi-linearity.

Proposition 4.38 *Let S be an* **A***-semi-linear set of $\mathbf{R}^n$, and let $\mathcal{D}$ be the algorithmic decomposition of $\overline{S}$ relative to S. Let S^0 be the union of all zero-dimensional members of $\mathcal{D}$. The following properties hold:*

1. *if S is bounded, then S is* $\mathbf{Z}$*-semi-linear if and only if all points in S^0 have rational coordinates;*
2. *if S is unbounded, and B is a* $\mathbf{Z}$*-semi-linear bounding box for S, then S is* $\mathbf{Z}$*-semi-linear if and only if $S \cap B$ is* $\mathbf{Z}$*-semi-linear; and*
3. *S is* $\mathbf{Z}$*-semi-linear if and only if all members of $\mathcal{D}$ are* $\mathbf{Z}$*-semi-linear.*

Proof. Let $\mathcal{T}$ be the set of the affine supports of the members of $\mathcal{D}$.

First, suppose that S is bounded.

We recall that Algorithm 4.33 can be implemented in FO + linear-$\mathbf{Z}$. Hence, if S is $\mathbf{Z}$-semi-linear, S^0 is $\mathbf{Z}$-semi-linear, too. Thus, all the points in S^0 must have rational coordinates.

Conversely, suppose that all points in S^0 have rational coordinates. Then S^0 is $\mathbf{Z}$-semi-linear. As in the proof of Lemma 4.17, it can then be shown that all members of $\mathcal{T}$ are $\mathbf{Z}$-semi-linear. As in the proof of Proposition 4.35, it can finally be shown that all members of $\mathcal{D}$, as well as S, are $\mathbf{Z}$-semi-linear.

Next, suppose that S is unbounded.

Of course, if S is $\mathbf{Z}$-semi-linear, then, since B is $\mathbf{Z}$-semi-linear, $S \cap B$ is $\mathbf{Z}$-semi-linear.

Conversely, suppose that $S_B = S \cap B$ is $\mathbf{Z}$-semi-linear. Let $\mathcal{D}_B$ be the algorithmic decomposition of $\overline{S \cap B}$ relative to $S \cap B$, and let S_B^0 be the union of all zero-dimensional members of $\mathcal{D}_B$. By Property 1, all points in S_B^0 have rational coordinates. From this, we deduce as above that all members of $\mathcal{D}_B$ are $\mathbf{Z}$-semi-linear. As in the proof of Lemma 4.32, we can show that, for each member of $\mathcal{D}$, there is a member of $\mathcal{D}_B$ with the same affine support. Thus, all members of $\mathcal{T}$ are $\mathbf{Z}$-semi-linear, from which we deduce as above that all members of $\mathcal{D}$, as well as S, are $\mathbf{Z}$-semi-linear. ∎

Proposition 4.38 yields a decidability criterion for $\mathbf{Z}$-semi-linearity, provided that rationality of a real algebraic number is decidable, which we show next.

Lemma 4.39 *Suppose a real algebraic number is given by a univariate polynomial equation with integer coefficients and an open interval with rational endpoints which contains that algebraic number as only solution of the equation. It is decidable whether that real algebraic number is rational.*

Proof. It is an easily provable consequence of Eisenstein's irreducibility criterion that any rational root of a polynomial $a_n x^n + \cdots + a_0$ with integer coefficients can be written as r/s with r and s relatively prime, $r|a_0$, and $s|a_n$. (This result is called

the *rational root theorem* in [65].) Hence, there are only a finite number of rational numbers for which the conditions defining the algebraic number have to be verified. ■

We can now finally prove our second main result of this chapter.

Theorem 4.40 *It is decidable whether a semi-algebraic set defined by a polynomial constraint formula is* **Z***-semi-linear.*

Proof. Let S be a semi-algebraic set of $\mathbf{R}^n$. First, we verify whether S satisfies Property SL. If S does not satisfy Property SL, it is not **R**-semi-linear (Proposition 4.2), whence certainly not **Z**-semi-linear; else it is **A**-semi-linear (Theorem 4.37).

If S is bounded, which can be decided in FO + linear-**Z**, we compute the set S^0 of all special points with Algorithm 4.18 applied to S. By Proposition 4.38, it now suffices to decide whether all points in S^0 have rational coordinates.

If S is unbounded, we apply Algorithm 4.33 to S to obtain the set S_B^0 of all special points of S. Notice that, if S is **Z**-semi-linear, the bounding box B used in Algorithm 4.33 is **Z**-semi-linear, too. By Proposition 4.38, it now suffices to decide whether all points in S_B^0 have rational coordinates. ■

However, the Boolean query deciding the **Z**-semi-linearity of a semi-algebraic set is *not* expressible in FO + poly.

Theorem 4.41 *The Boolean query of type* $[0, n] \to [0, 0]$ *deciding the* **Z***-semi-linearity of a semi-algebraic set is* not *expressible in* FO + poly*.*

Proof. Assume to the contrary that there exists a sentence σ in the first-order language $(\leq, S, +, \times, 0, 1)$, with S an n-dimensional predicate, such that, for each possible interpretation of S as a semi-algebraic set, σ is *true* if and only if this interpretation is a **Z**-semi-linear set of $\mathbf{R}^n$.

Now, let $x_1, \ldots, x_n$ be real variables not occurring in σ, and transform σ into a real constraint formula $\varphi(x_1, \ldots, x_n)$ by replacing each sub-formula $S(y_1, \ldots, y_n)$ in σ, with $y_1, \ldots, y_n$ variables bound in σ, by the sub-formula $(x_1 = y_1 \wedge \ldots \wedge x_n = y_n)$. Hence $\varphi(x_1, \ldots, x_n)$ evaluates to *true* if and only if the evaluation of $(x_1, \ldots, x_n)$ is a point with rational coordinates. Now, let $\psi(x)$ be the real constraint formula $\varphi(x, \ldots, x)$. Then $\{x \mid \psi(x)\}$ is the set of all rational numbers, which is not semi-algebraic, a contradiction. ■

Chapter 5

Extensions of FO + linear

In Chapter 3, we concluded that, despite several positive expressiveness results, FO + linear is not sufficiently powerful to accompany the linear constraint database model as a general-purpose linear query language. In this chapter, we focus on extensions of FO + linear which remain sound for the FO + poly^{lin} queries and are strictly more expressive than FO + linear.

First, we study a method to extend FO + linear with linear operators in a sound way.

Second, we experiment with adding multiplicative power to FO + linear in a sound way. The resulting linear query language is called PFOL. We characterize PFOL as an extension of FO + linear with two particular operators. To study the expressiveness of PFOL, independently of FO + linear, we develop a finite representation technique for arbitrary semi-linear sets and provide encoding and decoding algorithms expressible in PFOL. As a result, every PFOL-expressible query induces an equivalent PFOL-expressible query defined on the finite representations of the input and output of the original query, and, hence, the queries expressible in PFOL can be characterized in terms of the PFOL-expressible queries defined on finite representations. We then define in a syntactic way the query language SPFOL, which is guaranteed to yield finite outputs upon finite inputs, and we show that SPFOL captures precisely the PFOL-expressible queries which return finite outputs upon finite inputs. Next, we show that the expressive power of SPFOL and the linear query language SafeEuql [63] are closely related. The query language SafeEuql was designed to capture the ruler and compass constructions on finite point-sets in the

two-dimensional plane. We conclude that PFOL captures those linear queries for which the finite representation of the output can be "constructed" from the finite representation of the input.

Finally, we discuss syntactically defined query languages which are sound and complete for the FO+poly^{lin} queries and show how the existence of these query languages can be proved using results developed in this chapter and in Chapter 4. While none of the complete query languages we propose here can be seen as a "natural" query language for the FO + poly^{lin} queries, their existence at least shows that it is meaningful to search for more natural such languages.

5.1 FO + linear Extended with Operators

The basic idea is to extend FO + linear with certain *linear operators*, such as the linear queries listed in Theorem 3.37 or the collinearity or the convex-closure query.

However, we can*not* achieve our goal by adding the corresponding predicates to FO + linear. Indeed, from the proof of Proposition 3.36, it follows that, e.g., adding a predicate $\texttt{collinear}(\vec{x}, \vec{y}, \vec{z})$, which evaluates to *true* if its arguments are collinear points, would yield a language equivalent to FO + poly, as the product of real numbers would become definable. Moreover, the set $\{(\vec{x}, \vec{y}, \vec{z}) \mid \vec{x}, \vec{y}, \vec{z} \in \mathbf{R}^n \wedge \texttt{collinear}(\vec{x}, \vec{y}, \vec{z})\}$ is not a semi-linear subset of $\mathbf{R}^{3n}$. Obviously, we need a less liberal syntax to ensure that the extensions of FO + linear envisaged remain sound with respect to the FO + poly^{lin} queries.

The subtle point in the definition of our extension of FO + linear with operators, is that we disallow free *real* variables in set terms. Before we come to the formal definition of this new query language, we first explain what we mean by an "operator."

An *operator* is defined to be an FO + poly^{lin} query. The type of an operator is the type of the corresponding query. Hence, examples of operators of type $[0, n] \to [0, n]$ are the collinearity and the convex-closure query. Notice that, e.g., the collinearity *query* defines a mapping from semi-linear sets to semi-linear sets, i.e., a linear query, while the corresponding *predicate* defines a non-semi-linear algebraic set. We also mention that operators can be defined either as FO + poly$^{\mathbf{Z}\text{-}lin}$ or as FO + poly$^{\mathbf{a}\text{-}lin}$ queries, which yields the opportunity to define a linear query language for the **Z**-linear constraint model (FO + **Z**-linear extended with **Z**-linear operators) as well as the **A**-linear constraint model (FO + **A**-linear extended with **A**-linear operators). Unless stated otherwise, we do not make the distinction between the **Z**-linear and the **A**-linear case, and all results proposed will be valid in both cases.

Let $\mathcal{O}$ be a set of operator names O provided with a type, each of which represents

an operator op(O) of the same type.[1]

The query language FO + linear + $\mathcal{O}$ is then defined as an extension of FO + linear, as follows. First, we extend the terms of FO + linear with *set terms*:

- If φ is an FO + linear + $\mathcal{O}$ formula with m free value variables $v_1, \ldots, v_m$ and n free real variables $x_1, \ldots, x_n$, then, if $k \leq m$,

$$\{(v_1, \ldots, v_k; x_1, \ldots, x_n) \mid \varphi(v_1, \ldots, v_m; x_1, \ldots, x_n)\}$$

  is a *set term* of type $[k, n]$ with free value variables, $v_{k+1}, \ldots, v_m$ and *no* free real variables.[2]

- If O is an operator name in $\mathcal{O}$ of type $[m_1, n_1; \ldots; m_k, n_k] \to [m, n]$, and $S_1, \ldots, S_k$ are set terms of types $[m_1, n_1], \ldots, [m_k, n_k]$, respectively, then

$$O(S_1, \ldots, S_k)$$

  is a *set term* of type $[m, n]$ with free variables the free (value) variables in S_1 through S_k.

Finally, we extend the atomic formulae of FO + linear:

- Let S be a set term of type $[m, n]$, $v_1, \ldots, v_m$ value variables, and $x_1, \ldots, x_n$ real variables. Then, $S(v_1, \ldots, v_m; x_1, \ldots, x_n)$ is an *atomic formula* with free value variables $v_1, \ldots, v_m$, and the free variables of S, and with free real variables $x_1, \ldots, x_n$.

Semantically, when actual values are substituted for the free variables, a set term of type $[m, n]$ represents a subset of $D^m \times \mathbf{R}^n$. Now, consider an atomic formula of the form $S(v_1, \ldots, v_m; x_1, \ldots, x_n)$. When actual values are substituted for the free variables, this atomic formula evaluates to *true* if the evaluation of $(v_1, \ldots, v_m; x_1, \ldots, x_n)$ belongs to the set represented by the set term S. The full semantics of FO + linear + $\mathcal{O}$ is now straightforward to define.

The following soundness property is easily shown by structural induction:

Theorem 5.1 *The query language* FO + linear + $\mathcal{O}$ *only expresses* FO + poly^{lin}*-expressible queries.*

[1] To be practically relevant, the set $\mathcal{O}$ must be recursively enumerable.

[2] Observe that this definition allows us to interpret a predicate name R of type $[k, n]$ as a set term of type $[k, n]$.

The syntactic restriction that set terms do not contain free real variables is essential for Theorem 5.1 to hold.

Without going into details, we mention that it is possible to define an algebraic query language equivalent to FO + linear + $\mathcal{O}$ by extending the linear constraint algebra defined in Section 2.3 of Chapter 2 with the operators represented by $\mathcal{O}$. This equivalence result forms a theoretical justification for the approach Güting et al. have taken with the development of the ROSE-algebra [39, 40, 42, 44], which extends the relational algebra with a class of spatial operators.

We now illustrate our method for extending FO + linear with an example of an FO + linear + $\mathcal{O}$ query language in which we can express the *collinearity* and *convex-closure* queries described in Section 3.3, of which it was shown they are non-expressible in FO + linear.

Example 5.2 Let $\mathcal{O}$ be an infinite set of operator names $\texttt{segment}_n$ of type $[0, n] \to [0, n]$, $n \geq 0$. We then associate with each operator name $\texttt{segment}_n$ the operator $\mathrm{op}(\texttt{segment}_n)$ defined by

$$\mathrm{op}(\texttt{segment}_n)(S) = \{\vec{x} \in \mathbf{R}^n \mid (\exists \vec{y})(\exists \vec{z})(S(\vec{y}) \wedge S(\vec{z}) \wedge \vec{x} \in [\vec{y}, \vec{z}]\}),$$

for each semi-linear set S of $\mathbf{R}^n$. Thus, $\mathrm{op}(\texttt{segment}_n)(S)$ is the union of all closed line segments with both endpoints in S. The FO + linear + $\mathcal{O}$ formula

$$\underbrace{\texttt{segment}_n(\texttt{segment}_n(\ldots \texttt{segment}_n}_{n \text{ times}}(S) \cdots))(\vec{x})$$

computes the convex closure of S. Using the convex-closure query as a macro, the FO + linear + $\mathcal{O}$ formula

$$(\exists d)(\texttt{dim}_n(\{\vec{x} \mid \texttt{convexclosure}(S)(\vec{x})\}, d) \wedge d \leq 1)$$

expresses the collinearity query. ■

In the next section, we propose an entirely different extension of FO + **A**-linear which captures the queries constructible with ruler and compass. We show that this powerful extension of FO + **A**-linear can alternatively be captured by extending FO + **A**-linear with two **A**-linear operators, as a further illustration of the potential of this paradigm for extending FO + linear.

5.2 The Query Language PFOL

In this section, we present the query language PFOL. The main idea behind the introduction of this language is augmenting FO + linear with a limited amount of

multiplicative power, as is needed, for instance, to express collinearity or to compute convex closure or distances, without sacrificing the soundness of the language with respect to the class of linear queries. We show that PFOL is indeed sound with respect to the class of FO + poly-expressible **A**-linear queries, and illustrate its expressive power with some examples. In Section 5.3, we shall make a more profound study of the expressive power of the query language PFOL.

In order to define the linear query language PFOL, we first need to define an intermediate language FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$, where $D_1, \ldots, D_k$ are so-called domain symbols denoting finite sets. For this purpose, we assume two sorts of variables, called *real variables* and *product variables*. We shall use $x, y, z, \ldots$, possibly subscripted, to denote real variables, and $p, q, r, \ldots$, possibly subscripted, to denote product variables. Finally, we shall use the symbol t, possibly subscripted, to denote a variable that can either be a real variable or a product variable.

Definition 5.3 Let $D_1, \ldots, D_k$ be domain symbols. An atomic partial FO+linear+ $\mathrm{P}(D_1, \ldots, D_k)$ formula is defined as follows:

- $R(t_1, \ldots, t_n)$, with R a predicate symbol of type $[0, n]$ and $t_1, \ldots, t_n$ real variables or product variables, is an atomic partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula;
- $\sum_{i=1}^{n} a_i t_i \,\theta\, a$ with $a_1, \ldots, a_n$, and a real algebraic numbers, $t_1, \ldots, t_n$ real variables or product variables, and $\theta \in \{=, \neq, <, \leq, >, \geq\}$, is an atomic partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula;
- $t_1 = p\, t_2$, with t_1 and t_2 real variables or product variables and p a product variable, is an atomic partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula; and
- $t = \sqrt{|p|}$, with t a real variable or a product variable and p a product variable, is an atomic partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula.

A partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula is built from atomic partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formulae using the connectives $\neg$ and $\wedge$ and the quantifiers $(\exists x)$, with x a real variable, and $(\exists p \in D_i)$, with p a product variable and $1 \leq i \leq k$.

An FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula is a partial FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula without free product variables.

In the context of sets of real numbers as interpretations for $D_1, \ldots, D_k$ and a linear constraint database containing interpretations for the relevant predicate symbols, the semantics of an FO + linear + $\mathrm{P}(D_1, \ldots, D_k)$ formula is the obvious one. ■

Notice that, for $k = 0$, the above definition reduces to the definition of an FO + **A**-linear formula.

We now state an important soundness property of an FO + linear + P($D_1, \ldots, D_k$) formula.

Proposition 5.4 *Let $D_1, \ldots, D_k$ be domain symbols and let $\varphi(x_1, \ldots, x_n)$ be an* FO + linear + P($D_1, \ldots, D_k$) *formula with free variables $x_1, \ldots, x_n$. If $D_1, \ldots, D_k$ are interpreted as* finite *sets of real algebraic numbers, then the set $\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$ defined by φ is interpreted as an* **A***-semi-linear set.*

Proof. Let, for $i = 1, \ldots, k$, the interpretation of D_i be the finite set of real algebraic numbers $\{c_{i1}, \ldots, c_{im_i}\}$. We consecutively eliminate product variables in φ by replacing formulae of the form $(\exists p \in D_i)\psi$ by $\psi(c_{i1}/p) \vee \ldots \vee \psi(c_{im_i}/p)$ (or *false*, if D_i is interpreted as the empty set). Hence, if $\overline{\varphi}$, is the result of all the substitutions, then $\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\} = \{(x_1, \ldots, x_n) \mid \overline{\varphi}(x_1, \ldots, x_n)\}$. Since the latter set is expressed as the result of an FO + **A**-linear query on an **A**-linear constraint database, it is **A**-semi-linear. ∎

Observe that Proposition 5.4 is *not true* in the **Z**-linear setting, not even when the coefficients $a_1, \ldots, a_n, a$ in the atomic formulae $\sum_{i=1}^{n} a_i t_i \; \theta \; a$ of PFOL are restricted to be integers.

Using Definition 5.3, we now define the language PFOL.

Definition 5.5 A PFOL program is of the form

$$D_1 \leftarrow \varphi_1(x); \ldots; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\},$$

with, for $i = 1, \ldots, k$, D_i domain symbols, $\varphi_i(x)$ an FO + linear + P($D_1, \ldots, D_{i-1}$) formula, and $\varphi(x_1, \ldots, x_n)$ an FO + linear + P($D_1, \ldots, D_k$) formula.

The semantics of a PFOL program is as follows. First, $D_1, \ldots, D_k$ are consecutively interpreted as finite sets of real algebraic numbers. In this process, D_i is interpreted as the set $\{x \mid \varphi_i(x)\}$ if this set is finite, and as the empty set otherwise.[3] Next, the set $\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$ is interpreted in the obvious way. ∎

From Proposition 5.4, the following is immediate.

[3]Recall that it can be decided in FO + linear whether a semi-linear set is finite, as was shown in Section 3.1. By Proposition 5.4, the set $\{x \mid \varphi_i(x)\}$ is **A**-semi-linear. If, in addition, it is finite, it must therefore consist of real algebraic numbers.

Theorem 5.6 *The language* PFOL *is sound with respect to the* FO + poly$^{\mathbf{a}\text{-}lin}$ *queries.*

In other words, the limited multiplicative power that has been added to FO + **A**-linear did not destroy the soundness of the language with respect to linearity. To illustrate the gain in expressive power entailed by this limited addition of multiplicative power, we give examples of PFOL programs for queries known to be inexpressible in FO + **A**-linear (see Section 3.3).

Remark 5.7 In order to write down PFOL programs concisely, we remark that the syntax of an FO+linear+P$(D_1, \ldots, D_k)$ formula may be relaxed without increasing the expressive power of PFOL. In particular, on can allow atomic formulae of the form

$$\sum_{i=1}^{n} P_i(p_1, \ldots, p_m) t_i \; \theta \; P(p_1, \ldots, p_m),$$

with $P_1, \ldots, P_n, P$ expressions built from the product variables $p_1, \ldots, p_m$ and the real algebraic numbers, using addition, multiplication and absolute-value square rooting, and $t_1, \ldots, t_n$ product or real variables. We can also allow Boolean conjunctives other than "$\wedge$" and "$\neg$", and subformulae of the form $(\forall p \in D)\varphi$ as abbreviation of $\neg(\exists p \in D)\neg\varphi$. ■

In the following examples, the binary predicate symbol R represents a semi-linear set in the two-dimensional plane.

Example 5.8 The PFOL program

$$\begin{aligned}
&D_1 \leftarrow (\exists y)(R(x,y) \vee R(y,x));\\
&\{() \mid (\forall p \in D_1)(p \neq p) \vee\\
&\qquad (\exists p_x \in D_1)(\exists p_y \in D_1)(\exists q_x \in D_1)(\exists q_y \in D_1)\\
&\qquad (\forall r_x \in D_1)(\forall r_y \in D_1)(R(r_x, r_y) \Rightarrow\\
&\qquad (\exists \lambda)(r_x = p_x\lambda + q_x - q_x\lambda \wedge r_y = p_y\lambda + q_y - q_y\lambda))\}
\end{aligned}$$

decides whether the points stored in R are collinear, if R is finite, and returns the empty set otherwise. ■

Example 5.9 The PFOL program

$$\begin{array}{l} D_1 \leftarrow (\exists y)(R(x,y) \vee R(y,x)); \\ \{(x,y) \mid (\exists p_x \in D_1)(\exists p_y \in D_1)(\exists q_x \in D_1)(\exists q_y \in D_1) \\ \qquad (\exists r_x \in D_1)(\exists r_y \in D_1)(\exists \lambda_1)(\exists \lambda_2)(\exists \lambda_3) \\ \qquad (R(p_x,p_y) \wedge R(q_x,q_y) \wedge R(r_x,r_y) \wedge \\ \qquad \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \wedge \lambda_3 \geq 0 \wedge \\ \qquad \lambda_1 + \lambda_2 + \lambda_3 = 1 \wedge x = p_x\lambda_1 + q_x\lambda_2 + r_x\lambda_3 \wedge \\ \qquad y = p_y\lambda_1 + q_y\lambda_2 + r_y\lambda_3)\} \end{array}$$

computes the convex closure of the points stored in R, if R is finite, and the empty set otherwise.

The modification of the above PFOL program obtained by removing the conditions $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, and $\lambda_3 \geq 0$ computes the affine support of the points stored in R, if R is finite, and the empty set otherwise. ■

Example 5.10 The PFOL program

$$\begin{array}{l} D_1 \leftarrow (\exists y)(R(x,y) \vee R(y,x)); \\ \{(x) \mid (\exists p_x \in D_1)(\exists p_y \in D_1)(\exists q_x \in D_1)(\exists q_y \in D_1) \\ \qquad (R(p_x,p_y) \wedge R(q_x,q_y) \wedge \\ \qquad x = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2})\} \end{array}$$

computes the set of all distances between pairs of points stored in R, if R is finite, and the empty set otherwise. ■

One can also write PFOL programs deciding whether the points of an *arbitrary* semi-linear set of $\mathbf{R}^n$ are collinear, or computing the convex closure or the affine support of such a set. However, we shall not exhibit such programs, as their existence—as well as the way in which they can be obtained—will follow from Theorem 5.46 and its proof, further on in this chapter.

We conclude this section with a remark on the number of finite domains introduced in a PFOL program. One can wonder whether it can be avoided to introduce more than one finite domain in a PFOL program. Although this question is open, we believe that the use of several finite domains in a PFOL query contributes to a more natural language and cannot be avoided in some cases. To illustrate our point, consider the query which takes a finite $\mathbf{A}$-semi-linear set as input and computes, for each positive number in the input, the 4th root of that number. The most natural way to express this query in PFOL is as follows

$$\begin{array}{l} D_1 \leftarrow (\exists x)(R(x) \wedge x \geq 0); \\ D_2 \leftarrow (\exists p \in D_1)(x = \sqrt{|p|}); \\ \{(x) \mid (\exists p \in D_2)(x = \sqrt{|p|})\}. \end{array}$$

It is far from clear how to express this query in PFOL using just one finite domain.

5.3 Expressiveness of PFOL

In order to investigate the expressive power of a geometric query language on possibly infinite geometric databases, we can proceed as follows. First, we define a finite representation of the geometric databases which may occur as input or output of the query language under consideration. We then show that the encoding and decoding algorithms to obtain the finite representation of a geometric database and recompute the geometric database from its finite representation are expressible in the query language under consideration. If this has been accomplished, the expressive power of the query language can be characterized in terms of the expressive power of the fragment of the query language returning a finite representation as output on a finite representation as input. This idea has also been considered by Kuijpers et al. [63], in the context of SafeEuql, and by Benedikt and Libkin [10], on a more abstract level.

In this section, we first exhibit a finite representation for arbitrary **Z**-linear and **A**-linear constraint databases, together with encoding and decoding algorithms that are expressible in PFOL. The techniques used are derived from the properties of semi-linear sets described in Chapter 3 and Chapter 4. We next study the PFOL-expressible queries which return finite outputs on finite inputs. Thereto, we define a new syntactical query language, called SPFOL, and prove that SPFOL captures precisely the class of PFOL-expressible queries returning finite outputs on finite inputs. Then, we investigate the expressive power of the query language SPFOL. We show that SPFOL has the same expressive power as the query language SafeEuql [63], whence all ruler and compass constructions on finite sets of points in the plane can be expressed in SPFOL. This result gives a geometric justification to SPFOL. Finally, we discuss the expressive power of PFOL in terms of the expressive power of SPFOL.

5.3.1 Finite Representations of Semi-Linear Sets

As announced earlier, we present here an encoding and decoding algorithm, both implementable in PFOL, which compute a finite representation of an arbitrary semi-linear set and recompute the semi-linear set from its finite representation, respectively. We proceed as follows. First, we present the encoding and decoding algorithms, and define the canonical finite representation of a semi-linear set as the result of the encoding algorithm. Then, we define finite representations as a generalization of canonical finite representations in such a way that the decoding algorithm

is guaranteed to recompute the original semi-linear set on any finite representation of it. For clarity, we first consider the case where the linear constraint database consists of *bounded* semi-linear sets, and then say which modifications are required if unbounded semi-linear sets occur.

The Bounded Case

Let S be a bounded semi-linear set of $\mathbf{R}^n$. We describe the encoding. First, denote by T the set of special points of S, as computed by Algorithm 4.33. We recall Proposition 4.34 which states that any finite collection of hyperplanes such that every affine support of a subset of points of T can be described as an intersection of hyperplanes of that collection, partitions the n-dimensional space such that S is a finite union of cells of that partition. We now describe how such a collection of hyperplanes can be computed within PFOL. First, we determine the affine support of T, which can be done in PFOL (Example 5.9). We also determine the dimension of this affine support, say $k \leq n$, which can be done in FO+linear (see Theorem 3.6). Then, consider all possible sequences $\vec{p}_{11}, \ldots, \vec{p}_{1k}, \ldots, \vec{p}_{k1}, \ldots, \vec{p}_{kk}$ of k^2 points of T. Let, for $i = 1, \ldots, k$, A_i be the affine support of $\vec{p}_{i1}, \ldots, \vec{p}_{ik}$. Let U consists of all points $\vec{p}$ for which $\cap_{i=1}^{k} A_i = \{\vec{p}\}$. Clearly, $T \subseteq U$, and U can be computed from T in PFOL. (Where necessary, product variables can be introduced which range over the set of coordinates of points in T.)

Example 5.11 We recall Example 4.8 with S the two-dimensional semi-linear set shown in Figure 4.3 and T, the set of the special points of S, consisting of the three corner points of the outer triangle, the three corner points of the inner triangle, and the special point on the base of the outer triangle, 7 points in total. Clearly, the affine support of T is the two-dimensional plane itself, and hyperplanes are lines in the plane. All 6 lines shown in Figure 4.4 connect points of T, so the 12 points they generate (including the 7 points of T) all belong to U. The set U, however, contains many more points. For instance, two additional points are obtained by intersecting the line connecting the tops of both triangles with the two horizontal lines in Figure 4.4. ■

Next, we compute the finite relation S^{enc} with arity $n(n+1)$. The relation consists of $(n+1)$-tuples of n-dimensional points of U, not necessarily distinct, such that the open[4] convex closure of these points is contained in S. In particular, some tuples of S^{enc} denote a point of U which is contained in S, other tuples of S^{enc} denote two

[4] "Open" is to be understood with respect to the topology of the affine support of the points under consideration. In Example 5.9, the open convex closure can be obtained by requiring that λ_1, λ_2, and λ_3 are strictly greater than 0. This technique also works in general.

different points of U such that the open interval between these points is contained is S, still other tuples of S^{enc} denote three non-collinear points of U such that the open triangle of which these points are the corners is contained in S, and so on. Notice that tuples of S^{enc} consist of at most $k+1$ different points (k being the dimension of S). Since product variables can be used to represent the points of the finite relation U, and since the convex closure of a finite set of points can be computed in PFOL (Example 5.9), we conclude that the entire query of type $[n] \to [(n+1)n]$ which, given a semi-linear set S as input, returns the finite relation S^{enc} as output, can be computed in PFOL. The output relation S^{enc} is the *canonical finite representation* of S.

We consider the following property to be the key property of this canonical finite representation:

Proposition 5.12 *Let S be a bounded semi-linear set of $\mathbf{R}^n$ and let S^{enc} be the canonical finite representation of S. Then*

$$S = \bigcup_{(\vec{p}_1,\ldots,\vec{p}_{n+1})\in S^{enc}} \mathtt{CH}(\vec{p}_1,\ldots,\vec{p}_{n+1}),$$

where $\mathtt{CH}(\vec{p}_1,\ldots,\vec{p}_{n+1})$ is the open convex closure of $\vec{p}_1,\ldots,\vec{p}_{n+1}$.

Proof. By construction of the relation S^{enc}, the inclusion from right to left is trivial. Therefore, we concentrate on the other inclusion. From Proposition 4.34, we know that S is a finite union of cells of the partition of the affine support A of S induced by the $(k-1)$-dimensional hyperplanes supported by the points of T. By construction, the cells of this partition have their corner points in U. Since S is bounded, S is a finite union of bounded cells, and since these are intersections of half-planes and thus convex, they can be triangulated using their corner points only. Thus, each cell contained in S, whence S itself, is fully contained in the left-hand side of the above equality. ■

Decoding the canonical finite representation of a bounded semi-linear set can be done in PFOL in the obvious way, suggested by the formula in the statement of Proposition 5.12.

The General Case

Let S be a semi-linear set of $\mathbf{R}^n$ which may be unbounded. The methodology of the bounded case can easily be extended, provided we can incorporate into the finite representation the "corner points at infinity" of the unbounded cells of the

partition. "Corner points at infinity" will be represented by *directional vectors*. Hence, for each direction, we consider two "corner points at infinity," one for each orientation. To compute these directional vectors, it is necessary to introduce a bounding box B for $\overline{S}$ (recall Definition 4.20). One such bounding box can be computed with Algorithm 4.27, which, by Proposition 4.31, is implementable in FO + linear, whence in PFOL.

Thus, we construct in PFOL from $S \cap B$ the set T of special points. Let U be, in the same way as in the bounded case, those points $\vec{p}$ for which $\cap_{i=1}^{k} A_i = \{\vec{p}\}$ with k the dimension of the affine support of T and A_i the affine support of $k+1$ linear independent points of T. To find the "corner points at infinity," we compute the set V of directional vectors $\vec{s} = \pm(\vec{c} - \vec{p})$, where $\vec{c}$ is a point of T on the boundary of B and $\vec{p}$ is an arbitrary point of T, and the open interval between both points is fully contained in S[5]. Using the coordinates of the points of T as a finite domain for restricting the range of product variables, we can easily write a PFOL program to compute V.

The *canonical finite representation* of S is again a finite relation S^{enc}, but which is now of arity $(n+1)^2$. Intuitively, we add an extra coordinate to each point, which equals "1" for a "finite" corner point, and "0" for a "corner point at infinity," i.e., for a directional vector.

The relation S^{enc} is the subset of

$$\bigcup_{i=1}^{n+1} (U \times \{1\})^i \times (V \times \{0\})^{n+1-i},$$

with $1 \leq i \leq n+1$, consisting of all $n+1$-tuples

$$((\vec{p}_1, 1), \ldots, (\vec{p}_i, 1), (\vec{s}_{i+1}, 0), \ldots, (\vec{s}_{n+1}, 0))$$

such that the open convex closure of all these, not necessarily different, points and "directions" is contained in S. We notice that a point p is in this open convex closure if there exist $\lambda_1 > 0, \ldots, \lambda_i > 0, \mu_{j+1} > 0, \ldots, \mu_{n+1} > 0$ such that $\lambda_1 + \cdots + \lambda_i = 1$ and

$$\vec{p} = \lambda_1 \vec{p}_1 + \cdots + \lambda_i \vec{p}_i + \mu_{i+1} \vec{s}_{i+1} + \cdots + \mu_{n+1} \vec{s}_{n+1}.$$

Since product variables can be used to represent the points of the finite relations U and V, we conclude that the entire query of type $[n] \to [(n+1)^2]$, that, given a semi-linear set S as input, returns its canonical finite representation as output, can be computed in PFOL.

[5]The latter condition restricts the set of directional vectors to be considered, but will turn out not to be strictly necessary.

Example 5.13 Recall Example 4.29 with the two-dimensional set S shown with a bounding box B in Figure 4.7. For convenience, we shall assume that the angular point has coordinates $(0,0)$, while the special points on the bounding box have coordinates $(1,0)$ and $(1,0.5)$, respectively. The set T consists of those points. Clearly, $U = T$. The set V consists of 6 directional vectors, with coordinates $(1,0)$, $(-1,0)$, $(1,0.5)$, $(-1,-0.5)$, $(0,0.5)$, and $(0,-0.5)$, respectively. The canonical finite representation S^{enc} of S is the closure under permutations of generalized points within one tuple of the set

$$\{(0,0,1;0,0,1;0,0,1),(1,0,1;1,0,1;1,0,1),(1,\tfrac{1}{2},1;1,\tfrac{1}{2},1;1,\tfrac{1}{2},1),$$
$$(0,0,1;1,0,1;*,*,*),(0,0,1;1,\tfrac{1}{2},1;*,*,*),(1,0,1;1,\tfrac{1}{2},1;*,*,*),$$
$$(0,0,1;1,0,0;*,*,*),(0,0,1;1,\tfrac{1}{2},0;*,*,*),1,0,1;1,0,0;*,*,*),$$
$$(1,\tfrac{1}{2},1;1,\tfrac{1}{2},0;*,*,*),(0,0,1;1,0,1;1,\tfrac{1}{2},1),(1,0,1;1,\tfrac{1}{2},1;1,0,0),$$
$$(1,0,1;1,\tfrac{1}{2},1;1,\tfrac{1}{2},0),(0,0,1;1,0,1;1,\tfrac{1}{2},1),(0,0,1;1,\tfrac{1}{2},1;1,0,0),$$
$$(0,0,1;1,0,0;1,\tfrac{1}{2},0),(1,0,1;1,0,0;1,\tfrac{1}{2},0),(1,\tfrac{1}{2},1;1,0,0;1,\tfrac{1}{2},0)\},$$

with $(*,*,*)$ standing for one of the other points occuring in that tuple. ■

Proposition 5.12 can immediately be generalized:

Proposition 5.14 *Let S be an arbitrary semi-linear set of $\mathbf{R}^n$ and let S^{enc} be the canonical finite representation of S. Then*

$$S = \bigcup_{((\vec{p}_1,t_1),\ldots,(\vec{p}_{n+1},t_{n+1}))\in S^{enc}} \mathtt{CH}((\vec{p}_1,t_1),\ldots,(\vec{p}_{n+1},t_{n+1})),$$

where, for $i = 1,\ldots,n+1$, $t_i = 1$ or $t_i = 0$ and $\mathtt{CH}((\vec{p}_1,t_1),\ldots,(\vec{p}_{n+1},t_{n+1})) \subseteq \mathbf{R}^n$ is the open convex closure of $(\vec{p}_1,t_1),\ldots,(\vec{p}_i,t_i)$, where, for $i = 1,\ldots,n+1$, $(\vec{p}_i,t_i)$ is interpreted as the "finite" corner point $\vec{p}_i$, if $t_i = 1$, and as the "corner point at infinity" described by the directional vector $\vec{p}_i$, if $t_i = 0$.

Again, decoding the canonical finite representation of an arbitrary semi-linear set can be done in PFOL in the obvious way, suggested by the formula in the statement of Proposition 5.14.

We now define a *finite representation* of S as any relation of the right arity that satisfies Proposition 5.14. (For these relations, the *decoding* query described above always returns the original semi-linear set).

We now turn back to our motivation for considering finite representations, namely to study the expressive power of the query language PFOL. We illustrate how the

results on finite representations of semi-linear sets can be used to lift query languages defined on finite databases to query languages defined on linear constraint databases.

Let $\mathcal{Q}$ be a query language defined on finite databases. We define *Lift*($\mathcal{Q}$) to be the query language defined on arbitrary linear constraint databases consisting of all compositions of the PFOL encoding program, a query of $\mathcal{Q}$, and the PFOL decoding program. The following property is now immediate.

Theorem 5.15 *Let $\mathcal{P}$ be a linear query language defined on arbitrary semi-linear databases that is at least as expressive as* PFOL. *Let $\mathcal{Q}$ be a query language on finite databases whose expressive power is bounded by $\mathcal{P}$. Then Lift($\mathcal{Q}$) is a query language on arbitrary databases whose expressive power is bounded by $\mathcal{P}$. If, moreover, $\mathcal{Q}$ is complete for the $\mathcal{P}$-expressible queries from finite inputs to finite outputs, then Lift($\mathcal{Q}$) has the same expressive power as $\mathcal{P}$.*

To use Theorem 5.15 for our purposes, we need the following definition.

Definition 5.16 A PFOL-finite program is a PFOL program which returns finite outputs upon finite inputs. ■

Example 5.17 The PFOL programs in Examples 5.8 and 5.10 are examples of PFOL-finite programs. ■

Clearly, PFOL has the same expressive power as *Lift*(PFOL-finite). The language PFOL-finite is defined semantically, however, and, therefore, difficult to study. In the following subsection, we syntactically define a related language of PFOL, called SPFOL, which is "safe" with respect to finite databases, i.e, an SPFOL program applied on a finite database as input always yields a finite database as output. Moreover, we show that the language SPFOL and the language PFOL-finite are equivalent in expressive power. We then study the expressive power of the language SPFOL.

5.3.2 The Language SPFOL

We start with the definition of SPFOL.

Definition 5.18 Let $D_1, \ldots, D_k$ be domain symbols. An atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula and its set of free variables are defined as follows:

- `true` is an atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula, and its set of free variables is the empty set;

- `adom`(t), with t a real variable or a product variable, is an atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula, and its set of free variables equals $\{t\}$;

- $R(t_1, \ldots, t_n)$, with R a predicate symbol of type $[0, n]$ and $t_1, \ldots, t_n$ real variables or product variables, is an atomic partial safe FO+linear+P($D_1, \ldots, D_k$) formula, and its set of free variables equals $\{t_1, \ldots, t_n\}$;

- $Q_1(p_1, \ldots, p_n)\, x = Q_2(p_1, \ldots, p_n) \wedge Q_1(p_1, \ldots, p_n) \neq 0$, $p_1, \ldots, p_n$ product variables, x a real variable, and Q_1 and Q_2 expressions built from the product variables $p_1, \ldots, p_n$ and the real algebraic numbers using addition, multiplication, and absolute-value square rooting, is an atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula[6], and its set of free variables equals $\{x, p_1, \ldots, p_n\}$; and

- $Q(p_1, \ldots, p_n)\, \theta\, 0$, with $p_1, \ldots, p_n$ product variables and Q an expression built from the product variables $p_1, \ldots, p_n$ and the real algebraic numbers using addition, multiplication, and absolute-value square rooting, is an atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula, and its set of free variables equals $\{p_1, \ldots, p_n\}$.

A partial safe FO + linear + P($D_1, \ldots, D_k$) formula and its set of free variables are inductively defined as follows:

- an atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula with set of free variables $\mathcal{F}$ is a partial safe FO + linear + P($D_1, \ldots, D_k$) formula with set of free variables $\mathcal{F}$;

- if $\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r)$ is a partial safe FO+linear+P($D_1, \ldots, D_k$) formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r\}$, and $\psi(x_1, \ldots, x_m, q_1, \ldots, q_s)$ is a partial safe FO + linear + P($D_1, \ldots, D_k$) formula with set of free variables $\{x_1, \ldots, x_m, q_1, \ldots, q_s\}$, then

  $$\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r) \vee \psi(x_1, \ldots, x_m, q_1, \ldots, q_s)$$

  is a partial safe FO + linear + P($D_1, \ldots, D_k$) formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r, q_1, \ldots, q_s\}$;

[6]Obviously, if $Q_1(p_1, \ldots, p_n)$ is a constant $a \neq 0$, we abbreviate this atomic partial safe FO + linear + P($D_1, \ldots, D_k$) formula to $a\, x = Q_2(p_1, \ldots, p_n)$.

- if $\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r)$ is a partial safe FO+linear+P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r\}$, and $\psi(y_1, \ldots, y_n, q_1, \ldots, q_s)$ is a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula with set of free variables $\{y_1, \ldots, y_n, q_1, \ldots, q_s\}$, then

  $$\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r) \wedge \psi(y_1, \ldots, y_n, q_1, \ldots, q_s)$$

  is a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, y_1, \ldots, y_n, p_1, \ldots, p_r, q_1, \ldots, q_s\}$;

- if $\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r)$ is a partial safe FO+linear+P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r\}$, and $\psi(y_1, \ldots, y_n, q_1, \ldots, q_s)$ is a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula with set of free variables $\{y_1, \ldots, y_n, q_1, \ldots, q_s\}$, and $\{y_1, \ldots, y_n\} \subseteq \{x_1, \ldots, x_m\}$, then

  $$\varphi(x_1, \ldots, x_m, p_1, \ldots, p_n) \wedge \neg\psi(y_1, \ldots, y_n, q_1, \ldots, q_s)$$

  is a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r, q_1, \ldots, q_s\}$; and

- if $\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r)$ is a partial safe FO+linear+P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_r\}$, and if $1 \leq i \leq r$ and $1 \leq j \leq k$, then

  $$(\exists p_i \in D_j)\varphi(x_1, \ldots, x_m, p_1, \ldots, p_r)$$

  is a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula with set of free variables $\{x_1, \ldots, x_m, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_r\}$.

We define a safe FO + linear + P$(D_1, \ldots, D_k)$ formula as a partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula without free product variables.

In the context of sets of real numbers as interpretations for $D_1, \ldots, D_k$ and a linear constraint database containing interpretations for the relevant predicate symbols, the semantics of a safe FO + linear + P$(D_1, \ldots, D_k)$ formula is the obvious one, provided `adom` is interpreted as the active domain of the input database.

The syntax of an SPFOL program is the same as that of a PFOL program, except that all formulae are safe.

The semantics of an SPFOL program is defined in the same way as the semantics of a PFOL program. ■

Example 5.19 The PFOL program in Example 5.10 is actually an SPFOL program, provided D is defined using the active-domain atom. ■

The language SPFOL has the following basic properties:

Proposition 5.20 *1. Each constant linear query returning an input-independent finite output can be expressed in* SPFOL*.*

2. Each SPFOL *program is equivalent to a* PFOL *program.*

3. SPFOL *programs return finite outputs upon finite inputs.*

The first statement follows from the observation that the most obvious way to describe a finite relation yields a safe FO + linear formula. The second statement is obvious, whereas the third statement can be proved by structural induction.

Observe, as in the case of PFOL, that SPFOL programs do *not* necessarily return finite **Z**-semi-linear sets as output on **Z**-semi-linear sets as input, not even when the coefficients in the polynomials of the atomic formulae of SPFOL are restricted to be integer.

The remainder of this subsection is concerned with proving that, on finite inputs, the syntactically defined language SPFOL has the same expressive power as the semantically defined language PFOL-finite. The proof is built on a chain of partial results. Unless stated otherwise, all SPFOL-expressible queries considered below are implicitly assumed to take *finite* inputs only.

We first introduce the following notion.

Definition 5.21 *Let Q be a linear query. The* active range *of Q upon a possible input of Q is the set of all real numbers in the corresponding output of Q. The* active range query *is the query returning upon a possible input of Q the active range of Q. An* active range superset *of Q upon a possible input of Q is a superset of the active range of Q. An* active range superset query *of Q is any query returning upon a possible input of Q an active range superset of Q.* ■

We can now state our first partial result, which will turn out to be of key importance.

Lemma 5.22 *Let Q be an* FO + poly^{lin} *query returning finite outputs upon finite inputs. If an active range superset query of Q is expressible in* SPFOL*, then Q is expressible in* SPFOL*.*

Proof. Let $\varphi(x_1, \ldots, x_n)$ be an FO + poly formula computing Q. From a result by Benedikt and Libkin [8, 9], it follows that there exists an equivalent formula $\psi(x_1, \ldots, x_n)$ in which all quantified variables range over the active domain (of the

input of ψ). Without loss of generality, we assume that there are no universal quantifiers in ψ. Let D_{in} be a domain symbol corresponding to the active domain of the input. Let $\tilde{\psi}$ be ψ in which (i) each bound variable x has been replaced by a distinct product variable p_x, and (ii) each quantifier "$(\exists x)$" has been replaced by "$(\exists p_x \in D_{\text{in}})$" Finally, let

$$D_1 \leftarrow \vartheta_1(x); \ldots ; D_k \leftarrow \vartheta_k(x);$$
$$\{(x) \mid \vartheta(x)\}$$

be an SPFOL program computing an active range superset query of Q. Consider the "program"

$$D_{\text{in}} \leftarrow \texttt{adom}(x);$$
$$D_1 \leftarrow \vartheta_1(x); \ldots ; D_k \leftarrow \vartheta_k(x);$$
$$D_{\text{out}} \leftarrow \vartheta(x);$$
$$\{(x_1, \ldots, x_n) \mid (\exists p_1 \in D_{\text{out}}) \ldots (\exists p_n \in D_{\text{out}})(x_1 = p_1 \wedge \ldots \wedge x_n = p_n \wedge \tilde{\psi}(p_1, \ldots, p_n)\}.$$

It should first be observed that $\tilde{\psi}(p_1, \ldots, p_n)$ can be interpreted as a safe FO + linear + $\mathrm{P}(D_{\text{in}}, D_1, \ldots, D_k, D_{\text{out}})$ formula, because (i) all quantification is of the form "$(\exists p_x \in D_{\text{in}})$," and (ii) apart from quantification, the syntax of SPFOL does not impose restrictions on formulae in which only product variables occur. Hence the above "program" is an SPFOL program. Clearly, it expresses Q. ∎

The following result is immediate from Lemma 5.22:

Proposition 5.23 *The restriction to finite inputs of each* FO + poly*-expressible Boolean query is expressible in* SPFOL*.*

Lemma 5.24 *The active range query of an* SPFOL*-expressible query is* SPFOL*-expressible.*

Proof. Let Q be an SPFOL-expressible query. Then, clearly, the active range query of Q is expressible in FO + poly. By Lemma 5.22, it thus suffices to show that an active range superset query of Q is expressible in SPFOL. Let

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

be an SPFOL program for Q. Let $\psi(x_{i_1}, \ldots, x_{i_r}, p_1, \ldots, p_m)$, $1 \leq i_1, \ldots, i_r \leq n$ and $p_1, \ldots, p_m$ product variables, a partial safe subformula of $\varphi(x_1, \ldots, x_n)$ with set of free variables $\{x_{i_1}, \ldots, x_{i_r}, p_1, \ldots, p_m\}$. For $j = 1, \ldots, m$, let u_j, $1 \leq u_j \leq$

k, be the index such that p_j is quantified over D_{u_j} in φ. From the partial safe FO + linear + P$(D_1, \ldots, D_k)$ formula $\psi(x_{i_1}, \ldots, x_{i_r}, p_1, \ldots, p_m)$, we define the safe FO + linear + P$(D_1, \ldots, D_k)$ formula

$$\overline{\psi}(x_{i_1}, \ldots, x_{i_r}) \equiv (\exists p_1 \in D_{u_1}) \ldots (\exists p_m \in D_{u_m}) \psi(x_{i_1}, \ldots, x_{i_r}, p_1, \ldots, p_m).$$

Then, by structural induction, we prove, for each partial safe subformula

$$\psi(x_{i_1}, \ldots, x_{i_r}, p_1, \ldots, p_m)$$

of $\varphi(x_1, \ldots, x_n)$, that an active range superset query of the query expressed by the SPFOL program

$$D_1 \leftarrow \varphi_1(x); \ldots; D_k \leftarrow \varphi_k(x);$$
$$\{(x_{i_1}, \ldots, x_{i_r}) \mid \overline{\psi}(x_{i_1}, \ldots, x_{i_r})\}$$

can be expressed in SPFOL. We shall call the above program P_ψ.

If ψ is atomic, then either the empty set, $\{(x) \mid \mathtt{adom}(x)\}$, or P_ψ itself is an SPFOL program computing an active range super set query of P_ψ, settling the basis of the induction.

If ψ has the form $\psi_1 \vee \psi_2$, with ψ_1 and ψ_2 safe, then the query of which the output is the union of the output of an active range superset query of P_{ψ_1} and the output of an active range superset query of P_{ψ_2} is an active range superset query of P_ψ. (The validity of this claim hinges on the fact that ψ_1 and ψ_2 have the same free real variables.) Clearly, this union can be computed in SPFOL if the subqueries can be computed in SPFOL.

If ψ has the form $\psi_1 \wedge \psi_2$ or $\psi_1 \wedge \neg\psi_2$, with ψ_1 and ψ_2 safe, then an active range superset query of P_{ψ_1} is also an active range superset query of P_ψ.

Finally, if ψ has the form $(\exists p_i \in D_{u_i})\vartheta$, with ϑ safe, then $\overline{\psi} = \overline{\vartheta}$, whence an active range superset query of P_ϑ is trivially an active range superset query of $P_\psi = P_\vartheta$, concluding the induction step.

It now suffices to observe that $\overline{\varphi} = \varphi$, whence P_φ is an SPFOL program expressing Q. ∎

We now prove the following elementary property.

Proposition 5.25 *The* SPFOL*-expressible queries are closed under composition.*

Proof. Consider a composition

$$R_1 := Q_1 \,;\, \ldots \,;\, R_m := Q_m \,;\, Q(R_1, \ldots, R_m),$$

with $Q_1, \ldots, Q_m$, and Q SPFOL-expressible queries, and $R_1, \ldots, R_m$ the input predicates of Q. To compute this composition, the input predicates in Q must be substituted by the corresponding outputs of $Q_1, \ldots, Q_m$.

Let, for $i = 1, \ldots, m$,

$$D_{i1} \leftarrow \varphi_{i1}(x); \ldots; D_{ik_i} \leftarrow \varphi_{ik_i}(x); \\ \{(x_1, \ldots, x_{n_i}) \mid \varphi_i(x_1, \ldots, x_{n_i})\}$$

be an SPFOL program computing Q_i. Also, let, for $i = 1, \ldots, m$,

$$E_{i1} \leftarrow \vartheta_{i1}(x); \ldots; E_{il_i} \leftarrow \vartheta_{il_i}(x); \\ \{(x) \mid \vartheta_i(x)\}$$

be an SPFOL program expressing the active range query of Q_i (Lemma 5.24). Finally, let

$$D_1 \leftarrow \psi_1(x); \ldots; D_k \leftarrow \psi_k(x); \\ \{(x_1, \ldots, x_n) \mid \psi(x_1, \ldots, x_n)\}$$

be an SPFOL program computing Q. Without loss of generality, we may assume that all domain symbols are pairwise different (even if some of the queries involved are identical).

Let $\overline{\psi}_1, \ldots, \overline{\psi}_k$ and $\overline{\psi}$ be obtained from $\psi_1, \ldots, \psi_k$ and ψ, respectively, by the following substitutions:

1. each occurrence of "$R_i(t_1, \ldots, t_{n_i})$," $1 \leq i \leq m$, with $t_1, \ldots, t_{n_i}$ real variables or product variables, is replaced by "$\varphi_i(t_1, \ldots, t_{n_i})$"; and

2. each occurrence of "$\mathtt{adom}(t)$", with t a real variable or a product variable, is replaced by "$\vartheta_1(t) \vee \ldots \vee \vartheta_m(t)$".

Then,

$$\begin{array}{l} D_{11} \leftarrow \varphi_{11}(x); \ldots; D_{1k_1} \leftarrow \varphi_{1k_1}(x); \\ \qquad\qquad \vdots \\ D_{m1} \leftarrow \varphi_{m1}(x); \ldots; D_{mk_m} \leftarrow \varphi_{mk_m}(x); \\ E_{11} \leftarrow \vartheta_{11}(x); \ldots; E_{1l_1} \leftarrow \vartheta_{1l_1}(x); \\ \qquad\qquad \vdots \\ E_{m1} \leftarrow \vartheta_{m1}(x); \ldots; E_{ml_m} \leftarrow \vartheta_{ml_m}(x); \\ D_1 \leftarrow \overline{\psi}_1(x); \ldots; D_k \leftarrow \overline{\psi}_k(x); \\ \{(x_1, \ldots, x_n) \mid \overline{\psi}(x_1, \ldots, x_n)\} \end{array}$$

is an SPFOL program computing the composition under consideration. ∎

We shall make extensive use of Proposition 5.25 in the remainder of this section, however, without each time explicitly mentioning this.

The proof that each PFOL-finite program can be transformed into an equivalent SPFOL program will use structural induction. Unfortunately, intermediate results (i.e., outputs corresponding to subformulae of formulae in the program) of PFOL-finite programs may be infinite. Therefore, we intend to prove by structural induction that *finite representations of* these intermediate results can be computed in SPFOL. For this purpose, we prove the following lemmas.

Lemma 5.26 *Let S be a semi-linear set of $\mathbf{R}^n$. There exists an* SPFOL *program that, upon a finite representation of S as input, computes a finite representation of the complement $\mathbf{R}^n - S$ as output.*

Proof. Let $S^{enc} \subseteq \mathbf{R}^{(n+1)^2}$ be an arbitrary finite representation of S. From S^{enc}, we shall compute the corner points (both finite and infinite) of the cells of a partition $\mathcal{P}$ of $\mathbf{R}^n$ into convex cells such that, for every tuple of S^{enc}, the convex subset of $\mathbf{R}^n$ represented by that tuple is a union of cells of $\mathcal{P}$. Consequently, S, whence also $\mathbf{R}^n - S$, is a union of cells of $\mathcal{P}$. From the obtained corner points, we shall compute a finite representation of $\mathbf{R}^n - S$.

We show that the entire computation can be expressed in FO + poly, and that the set of the coordinates of the corner points, which is an active range superset of this query, can be computed in SPFOL. By Lemma 5.22, it then follows that the entire computation can be expressed in SPFOL.

Let T be the set of all (finite) corner points occurring in a tuple of S^{enc}, all points $\vec{p}+\vec{d}$, with $\vec{p}$ a (finite) corner point and $\vec{d}$ a directional vector occurring in a tuple of S^{enc}, the origin $\vec{0}$ of $\mathbf{R}^n$, and the unit vectors $\vec{e}_1, \ldots, \vec{e}_n$ of the canonical coordinate basis of $\mathbf{R}^n$. Clearly, T can be computed from S^{enc} in SPFOL.

Next, we perform the construction in Subsection 5.3.1 to obtain the set U consisting of the (finite) points that can be described as intersections of n $(n-1)$-dimensional hyperplanes supported by points of T. Let $\vec{p}_{ij} \equiv (p_{ij1}, \ldots, p_{ijn})$, $1 \leq i, j \leq n$, be points of T. The condition that $\vec{p}_{i1}, \ldots, \vec{p}_{in}$, $1 \leq i \leq n$, supports an $(n-1)$-dimensional hyperplane of $\mathbf{R}^n$ has the form $Q_i(p_{i11}, \ldots, p_{inn}) \neq 0$, with Q_i a polynomial in $p_{i11}, \ldots, p_{inn}$, expressing some determinant. To obtain the common point(s) of the n hyperplanes, we have to solve a system of equations

$$\begin{cases} Q_{11}(p_{111}, \ldots, p_{1nn})\, x_1 + \cdots + Q_{1n}(p_{111}, \ldots, p_{1nn})\, x_n &= Q_{10}(p_{111}, \ldots, p_{1nn}); \\ & \vdots \\ Q_{n1}(p_{n11}, \ldots, p_{nnn})\, x_1 + \cdots + Q_{nn}(p_{n11}, \ldots, p_{nnn})\, x_n &= Q_{n0}(p_{n11}, \ldots, p_{nnn}), \end{cases}$$

where, for $i = 1, \ldots, n$,

$$Q_{i1}(p_{i11}, \ldots, p_{inn})\, x_1 + \cdots + Q_{in}(p_{i11}, \ldots, p_{inn})\, x_n = Q_{i0}(p_{i11}, \ldots, p_{inn})$$

is an equation describing the hyperplane supported by $\vec{p}_{i1}, \ldots, \vec{p}_{in}$. This system has a unique solution if and only its determinant is nonzero, which is a condition of the form $Q(p_{111}, \ldots, p_{nnn}) \neq 0$, with Q a polynomial in $p_{111}, \ldots, p_{nnn}$. If this is the case, then the solution is given by equations of the form

$$Q(p_{111}, \ldots, p_{nnn})\, x_i = Q'_{i0}(p_{111}, \ldots, p_{nnn}),$$

with Q and $Q'_{10}, \ldots, Q'_{n0}$ polynomials in $p_{111}, \ldots, p_{nnn}$. Since, obviously, product variables can be used to represent coordinates of points of T, it follows that U can be computed from T in SPFOL.

Since $\vec{0}, \vec{e}_1, \ldots, \vec{e}_n$ are in T, the (finite) corner points occurring in a tuple of S^{enc} also occur in U. Hence, the partition of $\mathbf{R}^n$ into convex cells defined by T (of which the set of the (finite) corner points of its cells is precisely U) satisfies the requirements imposed at the beginning of this proof.

Next, let $V = \{\vec{p} - \vec{q} \mid \vec{p}, \vec{q} \in U \wedge \vec{p} \neq \vec{q}\}$. The set V is a set of directional vectors, which will be used to describe the infinite cells of $\mathcal{P}$. (Actually, V generally contains more directional vectors than needed, resulting in a refinement of $\mathcal{P}$, which is no objection).

Clearly, $U \cup V$ can be computed from T in SPFOL. Since the (finite) corner points of U and the directional vectors of V suffice to obtain a finite representation of $\mathbf{R}^n - S$, the query returning the set of the coordinates of the points of U and the points of V upon input S^{enc} is an active range superset query of the query we intend to express. By Lemma 5.24, this query is computable in SPFOL.

Finally, to obtain the finite representation of $\mathbf{R}^n - S$, we must select all tuples of $n + 1$ (finite and infinite) points of U and V (of which at least one belongs to U) for which the open convex closure does not meet any of the open convex closures of $n + 1$ (finite and infinite) points in a tuple of S^{enc}. Obviously, this query can be expressed in FO + poly. ∎

Lemma 5.27 *Let S_1 and S_2 be semi-linear sets of $\mathbf{R}^n$. There exist* SPFOL *programs that, upon finite representations of S_1 and S_2 as input, compute finite representations of the union $S_1 \cup S_2$, respectively the intersection $S_1 \cap S_2$, respectively the difference $S_1 - S_2$ as output.*

Proof. Let $S_1^{enc}, S_2^{enc} \subseteq \mathbf{R}^{(n+1)^2}$ be arbitrary finite representations of S_1 and S_2, respectively. Clearly, $S_1^{enc} \cup S_2^{enc}$ is a finite representation of $S_1 \cup S_2$, which can be computed from S_1^{enc} and S_2^{enc} in SPFOL.

Since intersection can be expressed in terms of union and complementation, and since difference can be expressed in terms of intersection and complementation, the above result and Lemma 5.26 together with Proposition 5.25 yields the second and the third claim of Lemma 5.27. ∎

Lemma 5.28 *Let S be a semi-linear set of $\mathbf{R}^n$. Let $i_1, \ldots, i_m$ be indices, not necessarily distinct and not necessarily in ascending order, such that $1 \leq i_1, \ldots, i_m \leq n$. There exists an* SPFOL *program that, upon a finite representations of S as input, computes a finite representation of the generalized projection*

$$\pi_{i_1,\ldots,i_m}(S) = \{(x_{i_1}, \ldots, x_{i_m}) \mid (x_1, \ldots, x_n) \in S\}$$

as output.

Proof. Let $S^{enc} \subseteq \mathbf{R}^{(n+1)^2}$ be an arbitrary finite representation of S. By definition,

$$S = \bigcup_{(\vec{x}_1,\ldots,\vec{x}_{n+1}) \in S^{enc}} \mathtt{CH}(\vec{x}_1, \ldots, \vec{x}_{n+1}).$$

Hence,

$$\pi_{i_1,\ldots,i_m}(S) = \bigcup_{(\vec{x}_1,\ldots,\vec{x}_{n+1}) \in S^{enc}} \mathtt{CH}(\pi_{i_1,\ldots,i_m,n+1}(\vec{x}_1), \ldots, \pi_{i_1,\ldots,i_m,n+1}(\vec{x}_{n+1})).$$

In these expressions, $\vec{x}_1, \ldots, \vec{x}_{n+1}$ are in $\mathbf{R}^n \times \{0,1\}$ and represent generalized points of $\mathbf{R}^n$; clearly, $\pi_{i_1,\ldots,i_m,n+1}(\vec{x}_1), \ldots, \pi_{i_1,\ldots,i_m,n+1}(\vec{x}_{n+1})$ are in $\mathbf{R}^m \times \{0,1\}$ and represent the generalized points of $\mathbf{R}^n$ that are the relevant projections of $\vec{x}_1, \ldots, \vec{x}_{n+1}$. Furthermore, if $\vec{y}_1, \ldots, \vec{y}_{n+1} \in \mathbf{R}^m \times \{0,1\}$ are generalized points of $\mathbf{R}^m$, then $\mathtt{CH}(\vec{y}_1, \ldots, \vec{y}_{n+1})$, which is topologically open in its affine support, is the union of all convex closures $\mathtt{CH}(\vec{y}_{j_1}, \ldots, \vec{y}_{j_{m+1}})$ for which (i) $1 \leq j_1, \ldots, j_{m+1} \leq n+1$ (notice that $j_1, \ldots, j_{m+1}$ are not necessarily all distinct or in ascending order), (ii) there is at least one finite point among $\vec{y}_{j_1}, \ldots, \vec{y}_{j_{m+1}}$, and (iii) $\mathtt{CH}(\vec{y}_{j_1}, \ldots, \vec{y}_{j_{m+1}}) \subseteq \mathtt{CH}(\vec{y}_1, \ldots, \vec{y}_{n+1})$. From these observations, it follows that a finite representation of $\pi_{i_1,\ldots,i_m}(S)$ can be computed from the relevant projections of the generalized points in S^{enc}. Obviously, this computation can be expressed in FO + poly. Moreover, the active domain of the input of this query, which is a primitive in SPFOL, is an active range superset of this query. By Lemma 5.22, it follows that this query can be expressed in SPFOL. ∎

Lemma 5.29 *Let S be a semi-linear set of $\mathbf{R}^n$. Let $m \geq 0$. There exists an* SPFOL *program that, upon a finite representation of S as input, computes a finite representation of $S \times \mathbf{R}^m$ as output.*

Proof. Because of Proposition 5.25, it suffices to consider the case $m = 1$ (Notice that $S \times \mathbf{R}^0 = S$). Let $S^{enc} \subseteq \mathbf{R}^{(n+1)^2}$ be an arbitrary finite representation of S. Then, for each tuple $(\vec{x}_1, i_1\,; \ldots ; \vec{x}_{n+1}, i_{n+1})$ of S^{enc},

$$\{(\vec{x}_1, 0, i_1\,; \ldots ; \vec{x}_{n+1}, 0, i_{n+1}\,; \underbrace{0, \ldots, 0}_{n \text{ times}}, \pm 1, 0), (\vec{x}_1, 0, i_1\,; \ldots ; \vec{x}_{n+1}, 0, i_{n+1}\,; \vec{x}_1, 0, i_1)\},$$

which is a subset of $\mathbf{R}^{(n+2)^2}$, is a finite representation of

$$\mathtt{CH}((\vec{x}_1, i_1), \ldots, (\vec{x}_{n+1}, i_{n+1})) \times \mathbf{R}.$$

(In the above expression, $(\vec{x}_j, i_j)$, $1 \leq j \leq n+1$ and $i_j \in \{0, 1\}$, is an element of $\mathbf{R}^{n+1}$ and represents a generalized point of $\mathbf{R}^n$; semi-colons have been inserted for clarity.) Clearly, a finite representation of $S \times \mathbf{R}$ can be computed from S^{enc} in FO + poly by making the union over all tuples of S^{enc} of the finite representations. Moreover, the active domain of the input of this query, which is a primitive in SPFOL, augmented with constants 0, +1, and −1, is an active range superset of this query. By Lemma 5.22, it follows that this query can be expressed in SPFOL. ■

Lemma 5.30 *Let S_1 be a semi-linear set of $\mathbf{R}^n$, and S_2 a semi-linear set of $\mathbf{R}^m$. There exists an* SPFOL *program that, upon finite representations of S_1 and S_2 as input, computes a finite representation of the Cartesian product $S_1 \times S_2 \subseteq \mathbf{R}^{n+m}$ as output.*

Proof. We have that $S_1 \times S_2 = S_1 \times \mathbf{R}^m \cap \mathbf{R}^n \times S_2$. The result now follows from Lemmas 5.27, 5.28, and 5.29 (notice that $\mathbf{R}^n \times S_2$ can be obtained from $S_2 \times \mathbf{R}^n$ by a generalized projection), together with Proposition 5.25. ■

Lemmas 5.27, 5.28, and 5.30 yield some more closure properties for SPFOL.

Lemma 5.31 *Let S be an arbitrary* finite *semi-linear set of $\mathbf{R}^n$, and let S^{enc} be the unique finite representation of S. The query mapping S to S^{enc} and the query mapping S^{enc} to S are both* SPFOL*-expressible.*

Proof. We start by observing that

$$S^{enc} = \{(\underbrace{\vec{x}, 1; \ldots ; \vec{x}, 1}_{n+1 \text{ times}}) \mid \vec{x} \in S\}.$$

Obviously, the query mapping S to S^{enc} and the query mapping S^{enc} to S are both FO + poly-expressible. For the former query, the active range equals the active

domain—which is an SPFOL primitive—augmented with the constant 1. Hence, the active range query is clearly SPFOL-expressible. For the query mapping S^{enc} to S, the active domain is an active range superset. By Lemma 5.22, it follows that both queries in the statement of Lemmma 5.31 are SPFOL-expressible. ■

The following is an immediate corollary to Lemmas 5.27, 5.28, 5.30, and 5.31, and Proposition 5.25.

Proposition 5.32 *The* SPFOL*-expressible queries are closed under union, intersection, difference, generalized projection, and Cartesian product.*

We are now ready for the proof that each PFOL-finite program can be transformed into an equivalent SPFOL program. The proof consists of a double induction, the inner one of which is exhibited in Lemma 5.33.

Lemma 5.33 *Let Q be a linear query returning finite outputs upon finite inputs, and let*

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, \ldots , x_n) \mid \varphi(x_1, \ldots , x_n)\}$$

be a PFOL *program computing Q. Assume that, for $i = 1, \ldots , k$, there exists an* SPFOL *program which computes the interpretation of the domain symbol D_i in the above program. Then there exists an* SPFOL *program computing Q.*

Proof. Let, for each product variable p that occurs in φ (as bound variable), $u(p)$, $1 \leq u(p) \leq k$, be the index such that p is quantified over $D_{u(p)}$. (Without loss of generality, we assume that each product variable occurs in only one quantification.) For each real variable x that occurs in φ (as bound or unbound variable), we put $u(x_i) = 0$. In the remainder of this proof, we interpret D_0 as $\mathbf{R}$. If $t_1, \ldots , t_r$ are real variables or product variables occurring in φ, then, by the assumption and by Lemmas 5.28, 5.29, 5.30, and 5.31, and Proposition 5.25, the query returning upon a possible input of Q a finite representation of $D_{u(t_1)} \times \ldots \times D_{u(t_r)}$ can be computed in SPFOL. Now, let $\psi(t_1, \ldots , t_r)$ be a subformula of $\varphi(x_1, \ldots , x_n)$ with set of free real and product variables $\{t_1, \ldots , t_r\}$. We define the *output* of $\psi(t_1, \ldots , t_r)$ as the query returning upon a possible (in particular, finite) input of Q the set

$$\{(t_1, \ldots , t_r) \mid \psi(t_1, \ldots , t_r) \wedge D_{u(t_1)}(t_1) \wedge \ldots \wedge D_{u(t_r)}(t_r)\}$$

as output. By structural induction, we show , for each such subformula $\psi(t_1, \ldots , t_r)$ of $\varphi(x_1, \ldots , x_n)$, that there exists an SPFOL program returning upon a possible input of Q a finite representation of the output of $\psi(t_1, \ldots , t_r)$.

For the basis of this induction, we first examine the atomic subformulae.

- $R(t_1, \ldots, t_r)$, *with* R *an input predicate symbol.* By definition, the output of $R(t_1, \ldots, t_r)$ is the intersection of R with $D_{u(t_1)} \times \ldots \times D_{u(t_r)}$. By Lemmas 5.27 and 5.31, and Proposition 5.25, it follows that a finite representation of the output of $R(t_1, \ldots, t_r)$ is also SPFOL-computable.

- $\sum_{i=1}^{r} a_i t_i \ \theta \ a$, *with* $a_1, \ldots, a_r$ *and* a *real algebraic numbers and* $\theta \in \{=, \neq, <, \leq, >, \geq\}$. The output of $\sum_{i=1}^{r} a_i t_i \ \theta \ a$ is the intersection of the set $\{(y_1, \ldots, y_r) \mid \sum_{i=1}^{r} a_i y_i \ \theta \ a\}$ with $D_{u(t_1)} \times \ldots \times D_{u(t_r)}$. The query returning the first set is a constant linear query (the output is input-independent). Therefore, the query returning some finite representation of this set is also a constant query, which, by Proposition 5.20, is SPFOL-computable. By Lemma 5.27 and Proposition 5.25, it follows that a finite representation of the output of $\sum_{i=1}^{r} a_i t_i \ \theta \ a$ is also SPFOL-computable.

- $t_1 = pt_2$, *with* p *a product variable.* First, consider the query Q_{init} which, upon input the interpretation of $D_{u(p)}$, returns the set $\{(y_1, y_2, y_3) \mid y_1 = y_3 y_2 \wedge D_{u(p)}(y_3)\}$. The partial output of this query corresponding to some fixed value of y_3 is a line in $\mathbf{R}^3$ through the points $(0, 0, y_3)$ and $(y_3, 1, y_3)$. Hence, the SPFOL program (using some vector notation for the purpose of abbreviation)

  $$D \leftarrow D_{u(p)}(x);$$
  $$\{(\vec{z}_1, \vec{z}_2, \vec{z}_3, \vec{z}_4) \mid (\exists p \in D)(\vec{z}_1 = (0, 0, p, 1) \wedge \vec{z}_2 = (p, 1, 0, 0) \wedge \vec{z}_3 = (-p, -1, 0, 0) \wedge \vec{z}_4 = \vec{z}_3)\}$$

  computes, upon input the interpretation of $D_{u(p)}$, a finite representation of the output of Q_{init}. By Proposition 5.25, it follows that the query which, upon input a possible input of Q, returns a finite representation of Q_{init}, is also SPFOL-computable. Since the output of $t_1 = pt_2$ is the intersection of the output of Q_{init} with $D_{u(t_1)} \times D_{u(t_2)} \times D_{u(p)}$, it follows from Lemma 5.27 that a finite representation of the output of $t_1 = pt_2$ is also SPFOL-computable.

- $t = \sqrt{|p|}$, *with* p *a product variable.* First, consider the query Q_{init} which, upon input the interpretation of $D_{u(p)}$, returns the finite set $\{(y_1, y_2) \mid y_1 = \sqrt{|y_2|} \wedge D_{u(p)}(y_2)\}$. This query is computed by the SPFOL program

  $$D \leftarrow D_{u(p)}(x);$$
  $$\{(y_1, y_2) \mid (\exists p \in D)(y_1 = \sqrt{|p|} \wedge y_2 = p)\}.$$

  Since the output of Q_{init} is always finite, it follows from Lemma 5.31 that the query returning a finite representation of the output of Q_{init} is also SPFOL-computable. Since the output of $t = \sqrt{|p|}$ is the intersection of the output of Q_{init} with $D_{u(t)} \times D_{u(p)}$, it follows from Lemma 5.27 that a finite representation of the output of $t = \sqrt{|p|}$ is also SPFOL-computable.

This concludes the basis of the induction. For the induction step, we consider the ways in which larger formulae can be constructed.

- *Boolean combination.* It suffices to consider the connectives "$\neg$" and $\wedge$." First, let $\neg\psi(t_1, \ldots, t_r)$ be a subformula of $\varphi(x_1, \ldots, x_n)$. Clearly, if a finite representation of the output of $\psi(t_1, \ldots, t_r)$ is SPFOL-computable, then, by Lemma 5.26 and Proposition 5.25, a finite representation of the output of $\neg\psi(t_1, \ldots, t_r)$ is also SPFOL-computable. Next, let $\psi_1(t_1, \ldots, t_v, \ldots, t_s) \wedge \psi_2(t_v, \ldots, t_s, \ldots, t_r)$ be a subformula of $\varphi(x_1, \ldots, x_n)$. Without loss of generality, we assume that $t_v, \ldots, t_s$ are the free (real and product) variables shared by ψ_1 and ψ_2. If $S_1 \subseteq \mathbf{R}^s$ and $S_2 \subseteq \mathbf{R}^{r-v+1}$ are the outputs of $\psi_1(t_1, \ldots, t_s)$ and $\psi_2(t_v, \ldots, t_r)$, then $S_1 \times \mathbf{R}^{r-s} \cap \mathbf{R}^{v-1} \times S_2$ is the output of $\psi_1(t_1, \ldots, t_v, \ldots, t_s) \wedge \psi_2(t_v, \ldots, t_s, \ldots, t_r)$. Hence, if finite representations of the outputs of $\psi_1(t_1, \ldots, t_s)$ and $\psi_2(t_v, \ldots, t_r)$ are SPFOL-computable, then, by Lemmas 5.27, 5.28, and 5.29, and Proposition 5.25, a finite representation of the output of $\psi_1(t_1, \ldots, t_v, \ldots, t_s) \wedge \psi_2(t_v, \ldots, t_s, \ldots, t_r)$ is SPFOL-computable.

- *Quantification.* If $(\exists x)\psi(t_1, \ldots, t_r)$, with $x \in \{t_1, \ldots, t_r\}$ a real variable, is a subformula of $\varphi(x_1, \ldots, x_n)$, then the output of $(\exists x)\psi(t_1, \ldots, t_r)$ is obtained from the output of $\psi(t_1, \ldots, t_r)$ by projecting out the coordinate position corresponding to the free real variable x of ψ. Similarly, if $(\exists p \in D_{u(p)})\psi(t_1, \ldots, t_r)$, with $p \in \{t_1, \ldots, t_r\}$ a product variable, is a subformula of $\varphi(x_1, \ldots, x_n)$, then the output of $(\exists p \in D_{u(p)})\psi(t_1, \ldots, t_r)$ is obtained from the output of $\psi(t_1, \ldots, t_r)$ by projecting out the coordinate position corresponding to the free product variable p of ψ. Hence, if a finite representation of the output of $\psi(t_1, \ldots, t_r)$ is SPFOL-computable, then, by Lemma 5.28, a finite representation of the output of $(\exists x)\psi(t_1, \ldots, t_r)$, respectively $(\exists p \in D_{u(p)})\psi(t_1, \ldots, t_r)$, is SPFOL-computable.

We have thus shown by structural induction that, for each subformula $\psi(t_1, \ldots, t_r)$ of $\varphi(x_1, \ldots, x_n)$, there exists an SPFOL program returning upon a possible input of the given query Q a finite representation of the output of $\psi(t_1, \ldots, t_r)$. By definition, the output of $\varphi(x_1, \ldots, x_n)$ is the output of Q. Hence we have shown that, in particular, a finite representation of the output of Q is SPFOL-computable. Since, by assumption, the output of Q is always finite, it follows from Lemma 5.31 that Q itself is SPFOL-computable. ∎

Theorem 5.34 *The languages* PFOL-finite *and* SPFOL *are equivalent in expressive power.*

Proof. By Proposition 5.20, each SPFOL program is equivalent to a PFOL-finite program. We are going to show that each PFOL-finite program is equivalent to an SPFOL program, by induction on the number of domain symbols in the PFOL-finite program.

For the basis of this induction, we observe that a PFOL-finite program without domain symbols satisfies all the conditions of Lemma 5.33, whence such a program is equivalent to an SPFOL program.

For the induction step, let $k > 0$, and assume as induction hypothesis that all PFOL-finite programs with at most $k-1$ domain symbols are equivalent to SPFOL programs. Let

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, \ldots , x_n) \mid \varphi(x_1, \ldots , x_n)\},$$

be a PFOL-finite program with k domain symbols, which we shall call P. For $i = 1, \ldots , k$, let

$$\overline{\varphi}_i(x) \equiv \varphi_i(x) \wedge (\forall x)(\exists \varepsilon)(\forall y)(\varepsilon > 0 \wedge \varphi_i(x) \wedge y \neq x \wedge x - \varepsilon \leq y \leq x + \varepsilon \;\Rightarrow\; \neg\varphi(y)).$$

Since $\varphi_i(x)$ is an FO + linear + $\mathrm{P}(D_1, \ldots , D_{i-1})$ formula, $\overline{\varphi}_i(x)$ is also an FO + linear + $\mathrm{P}(D_1, \ldots , D_{i-1})$ formula. Clearly, $\{(x) \mid \overline{\varphi}_i(x)\}$ equals $\{(x) \mid \varphi_i(x)\}$, if this set is finite, and the empty set, otherwise. Hence, we may conclude that

$$D_1 \leftarrow \overline{\varphi}_1(x); \ldots ; D_{i-1} \leftarrow \overline{\varphi}_{i-1}(x);$$
$$\{(x) \mid \overline{\varphi}_i(x)\},$$

is a PFOL-finite program with $i - 1 < k$ domain symbols computing the interpretation of D_i in P. By the induction hypothesis, there exists an equivalent SPFOL program. We have thus shown that, for $i = 1, \ldots , k$, there exists an SPFOL program which computes the interpretation of the domain symbol D_i in P. Another application of Lemma 5.33 yields the existence of an SPFOL program equivalent to P. ∎

If `encode` and `decode` are the PFOL programs computing the canonical finite representation of a semi-linear set, respectively re-computing a semi-linear set from any of its finite representations, and P is an arbitrary PFOL program, then the composition $\texttt{encode} \circ P \circ \texttt{decode}$, which represents a PFOL-finite program, computes, upon as input an arbitrary finite representation of the input of P, as output the canonical finite representation of the output of P. Corollary 5.35 is now immediate.

Corollary 5.35 *For every* PFOL *program P, there exists an* SPFOL *program which computes, upon as input an arbitrary finite representation of the input of P, as output the canonical finite representation of the output of P.*

Corollary 5.35 also yields a normal form for PFOL-expressible queries: each PFOL-expressible query is the composition of the `encode` program, followed by an SPFOL program, followed by the `decode` program.

5.3.3 Expressiveness of SPFOL

Although SPFOL is a coordinate-based language, we claim that it is nevertheless geometrically meaningful. To substantiate this claim, we prove that SPFOL programs in which the coefficients of the polynomials in atomic formulae are restricted to rational (or, equivalently, integer) numbers (instead of real algebraic numbers), define the same queries in the two-dimensional plane as the query language SafeEuql [63]. Indeed, SafeEuql has been devised to capture precisely the ruler and compass constructions on finite sets of points in the two-dimensional plane which return finite sets of points[7]. We take the liberty of redefining SafeEuql here, so that the definition may better accommodate proofs by structural induction.

We assume that the points $\vec{0}$, $\vec{e}_1$, and $\vec{e}_2$ represent the origin and both unit coordinate vectors of the canonical coordinate system of the two-dimensional plane. Variables, denoted as $\vec{p}, \vec{q}, \vec{r}, \ldots$, and possibly subscripted, should be interpreted as points of the plane rather than coordinate vectors. Predicates, denoted as $\hat{R}(\vec{p}_1, \ldots, \vec{p}_k)$, and possibly subscripted, represent relations consisting of k-ary tuples of points.

Definition 5.36 The SafeEuql formulae are the smallest collection of formulae closed under the following rules:

- `true` is a safe formula;
- $\hat{R}(\vec{p}_1, \ldots, \vec{p}_k)$ is a safe formula;
- if $\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ is a safe formula, and if $\{\vec{q}_1, \ldots, \vec{q}_6\} \subseteq \{\vec{p}_1, \ldots, \vec{p}_k\}$, and if $\vec{r}$ is an arbitrary variable, then
  - $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \vec{r} = \vec{0}$,

[7] A point of the plane is constructible with ruler and compass from a *finite* set of points if and only if that point can be obtained by applying a finite number of ruler and compass constructions on the given set of points and already constructed points. The following constructions can be executed with ruler and compass on a finite set of points of the plane:

1. the construction of a line through two points of the given set;
2. the construction of a circle of which the midpoint and a point belong to the given set;
3. the construction of the intersection point(s) of two lines, two circles, or a line and a circle.

  - $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \vec{r} = \vec{e}_1$,
  - $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \vec{r} = \vec{e}_2$,
  - $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \vec{r} = \vec{q}_1$,
  - $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \texttt{geom-pred}(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$,

  with `geom-pred` one of the geometric predicates defined below, are safe formulae;

- if $\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ and $\psi(\vec{p}_1, \ldots, \vec{p}_k)$ are safe formulae, then $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \vee \psi(\vec{p}_1, \ldots, \vec{p}_k)$ is a safe formula;
- if $\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ and $\psi(\vec{q}_1, \ldots, \vec{q}_l)$ are safe formulae, then $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \psi(\vec{q}_1, \ldots, \vec{q}_l)$ is a safe formula;
- if $\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ and $\psi(\vec{q}_1, \ldots, \vec{q}_l)$ are safe formulae such that $\{\vec{q}_1, \ldots, \vec{q}_l\} \subseteq \{\vec{p}_1, \ldots, \vec{p}_k\}$, then $\varphi(\vec{p}_1, \ldots, \vec{p}_k) \wedge \neg\psi(\vec{q}_1, \ldots, \vec{q}_l)$ is a safe formula; and
- if $\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ is a safe formula, and $1 \leq i \leq k$, then $(\exists \vec{p}_i)\varphi(\vec{p}_1, \ldots, \vec{p}_k)$ is a safe formula.

The semantics of a SafeEuql formula is the obvious one, provided the interpretation of `geom-pred` is one of the following geometric predicates[8]:

- `on-line`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_2$ and $\vec{q}_3$ are distinct and $\vec{q}_1$, $\vec{q}_2$, and $\vec{q}_3$ are collinear;
- `on-same-side`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_3$ and $\vec{q}_4$ are distinct, and $\vec{q}_1$ and $\vec{q}_2$ are on the same side of the line defined by $\vec{q}_3$ and $\vec{q}_4$;
- `on-circle`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_3$ and $\vec{q}_4$ are distinct and $\vec{q}_1$ is on the circle with center $\vec{q}_2$ and radius the distance between $\vec{q}_3$ and $\vec{q}_4$;
- `in-circle`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_3$ and $\vec{q}_4$ are distinct, and $\vec{q}_1$ is in the topological interior of the disk with center $\vec{q}_2$ and radius the distance between $\vec{q}_3$ and $\vec{q}_4$;
- `l-order`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_1$, $\vec{q}_2$, and $\vec{q}_3$ are collinear and distinct, and $\vec{q}_2$ is between $\vec{q}_1$ and $\vec{q}_3$;
- `c-order`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_1, \ldots, \vec{q}_4$ are pairwise distinct and on the same circle in clockwise or counter-clockwise order;

[8] Some variables serve as dummy variables to make geometric predicates of equal width.

- `l-l-crossing`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_1$ and $\vec{q}_2$, respectively $\vec{q}_3$ and $\vec{q}_4$ are distinct, $\vec{q}_1, \ldots, \vec{q}_4$ are not collinear, and $\vec{r}$ is the intersection point of the line through $\vec{q}_1$ and $\vec{q}_2$, and the line through $\vec{q}_3$ and $\vec{q}_4$;
- `l-c-crossing`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_1$ and $\vec{q}_2$, respectively $\vec{q}_4$ and $\vec{q}_5$ are distinct, and $\vec{r}$ is an intersection point of the line through $\vec{q}_1$ and $\vec{q}_2$, and the circle with center $\vec{q}_3$ and radius the distance between $\vec{q}_4$ and $\vec{q}_5$;
- `c-c-crossing`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_6)$, which evaluates to *true* if $\vec{q}_2$ and $\vec{q}_3$, respectively $\vec{q}_5$ and $\vec{q}_6$ are distinct, and $\vec{r}$ is an intersection point of the circle with center $\vec{q}_1$ and radius the distance between $\vec{q}_2$ and $\vec{q}_3$, and the circle with center $\vec{q}_4$ and radius $\vec{q}_5$ and $\vec{q}_6$. ■

Example 5.37 To illustrate Definition 5.36, we give an example of a SafeEuql query involving the geometric predicate `l-l-crossing`$(\vec{r}, \vec{q}_1, \ldots, \vec{q}_4)$.

Let $\hat{R}$ be a finite set of points in the two-dimensional plane. The SafeEuql query

$$\{(\vec{x}) \mid (\exists\vec{p})(\exists\vec{q})(\exists\vec{r})(\exists\vec{s})(\exists\vec{t})(\exists\vec{u})(\hat{R}(\vec{p}) \wedge \hat{R}(\vec{q}) \wedge \hat{R}(\vec{r}) \wedge \hat{R}(\vec{s}) \wedge \hat{R}(\vec{t}) \wedge \hat{R}(\vec{u}) \wedge \texttt{l-l-crossing}(\vec{x}, \vec{p}, \vec{q}, \vec{r}, \vec{s}, \vec{t}, \vec{u})\}$$

computes the intersection points of all pairs of different non-parallel lines supported by points of $\hat{R}$. ■

For a comparison between SPFOL and SafeEuql to make sense, we must realize that (i) expressions in atomic formulae of SPFOL are allowed to have real algebraic coefficients, and (ii) SPFOL works with variables representing real numbers, whereas SafeEuql works with variables representing points in the two-dimensional plane. Clearly, as SafeEuql expresses precisely the ruler and compass constructions, only *constructible numbers*[9] can be computed in SafeEuql. Observe that not every algebraic number can be constructed with ruler and compass from 0 and 1. To show this, we consider the following well-known result.

[9]With every point in the plane, we can associate a pair of real numbers corresponding to the coordinates of that point with respect to the canonical coordinate system of the plane. We then define the *constructible numbers* as $x \in \mathbf{R}$ such that the point with coordinates $(x, 0)$ can be constructed with ruler and compass from the points with coordinates $(0, 0)$ and $(1, 0)$. It is well-known that, for a and b real numbers, $a + b$, ab, $-a$, and $1/a$ can be constructed with ruler and compass from 0 and 1 (see Figure 3.6 and Figure 5.1). Hence, starting from 0 and 1, all rational numbers are constructible with ruler and compass. Moreover, the absolute-value square root of a given number can be constructed with ruler and compass (see also Figure 3.6), whence the constructible numbers are a strict superset of the rational numbers. The following alternative characterization of the constructible numbers can be found in the literature: a real number x is constructible with ruler and compass from 0 and 1 if and only if x belongs to a finite Galois-extension of $\mathbf{Q}$ with degree 2^n, $n \in \mathbf{N}$.

Proposition 5.38 *In three-dimensional space, the cube with volume twice the volume of a given cube cannot be constructed by ruler and compass.*

It follows from Proposition 5.38 that the real algebraic number $\sqrt[3]{2}$ is not a constructible number. We shall, therefore, only consider SPFOL programs in which the expressions in atomic formulae are built from product variables and *rational numbers*, using addition, multiplication, and absolute-value square rooting. We denote this restriction of SPFOL as SPFOL$^{\mathbf{Q}}$. Moreover, we assume from now on that all relation names involved, as well as the output, have even arity. This latter assumption allows us to interpret consecutive variables in a relation name, or in the output format, as coordinates of a point in the two-dimensional plane.

We first show that every SafeEuql-expressible query can be expressed in SPFOL$^{\mathbf{Q}}$. Thereto, we need the following lemma.

Lemma 5.39 *Let R be a finite relation of type $[0,2]$, i.e., a finite set of two-dimensional points. Let* `geom-pred`$(\vec{p}, \vec{p}_1, \ldots, \vec{p}_6)$ *be one of the geometric predicates* `l-l-crossing`*,* `l-c-crossing`*, and* `c-c-crossing` *defined in Definition 5.36. There exists an* SPFOL$^{\mathbf{Q}}$ *program of which the output upon input R equals the set*

$$\{(\vec{p}) \mid (\exists \vec{p}_1) \ldots (\exists \vec{p}_6) \hat{R}(\vec{p}_1) \wedge \ldots \wedge \hat{R}(\vec{p}_6) \wedge \texttt{geom-pred}(\vec{p}, \vec{p}_1, \ldots, \vec{p}_6)\}.$$

Proof. We describe, for every geometric predicate, how to obtain an SPFOL$^{\mathbf{Q}}$ program that has the desired point-set as output upon input R.

- *The geometric predicate* `l-l-crossing`$(\vec{p}, \vec{p}_1, \ldots, \vec{p}_6)$. First, we observe that the Boolean conditions "$\vec{p}_1$ and $\vec{p}_2$ are distinct", "$\vec{p}_3$ and $\vec{p}_4$ are distinct," and "$\vec{p}_1, \ldots, \vec{p}_4$ are not collinear," with $\vec{p}_1, \ldots, \vec{p}_4$ points of R, can be expressed in FO + poly, whence, by Proposition 5.23, they can be also expressed in SPFOL$^{\mathbf{Q}}$ [10]. Next, the intersection point $\vec{p}$ is found as the unique solution of the system of equations[11]

$$\begin{cases} (p^y - p_1^y)(p_2^x - p_1^x) - (p^x - p_1^x)(p_2^y - p_1^y) = 0 \\ (p^y - p_3^y)(p_4^x - p_3^x) - (p^x - p_3^x)(p_4^y - p_3^y) = 0 \end{cases}$$

  Solving this system of equations for p^x and p^y yields equations of the form $A_1 p^x = A_2$ and $B_1 p^y = B_2$, with $A_1 \neq 0$, A_2, $B_1 \neq 0$, and B_2 polynomials in

[10] Indeed, an inspection of the proof of Lemma 5.22 shows that, if the active range superset query of an FO + poly-expressible query Q can be expressed in SPFOL$^{\mathbf{Q}}$, the query Q can be also expressed in SPFOL$^{\mathbf{Q}}$. Hence, Proposition 5.23 remains valid for SPFOL$^{\mathbf{Q}}$.

[11] We shall denote the coordinates of a two-dimensional point $\vec{p}$ as p^x and p^y.

the variables $p_1^x, p_1^y, \ldots, p_4^x, p_4^y$. Hence, if D is the finite domain containing the active domain of R, all computations can be done with product variables ranging over domain D, whence there exists an SPFOL$^{\mathbf{Q}}$ program that computes the intersection point of two non-parallel lines. By Proposition 5.25, there exists an SPFOL$^{\mathbf{Q}}$ program that, upon input R, has the desired point-set as output.

- *The geometric predicate* `l-c-crossing`$(\vec{p}, \vec{p}_1, \ldots, \vec{p}_6)$. First, we observe that the Boolean conditions "$\vec{p}_1$ and $\vec{p}_2$ are distinct," and "$\vec{p}_4$ and $\vec{p}_5$ are distinct," with $\vec{p}_1, \vec{p}_2, \vec{p}_4, \vec{p}_5$ points of R, can be expressed in FO + poly, whence, by Proposition 5.23, they can be also expressed in SPFOL$^{\mathbf{Q}}$. Next, the intersection point $\vec{p}$ has to satisfy the following system of equations:

$$\begin{cases} (p^y - p_1^y)(p_2^x - p_1^x) = (p^x - p_1^x)(p_2^y - p_1^y) \\ (p^x - p_3^x)^2 + (p^y - p_3^y)^2 = (p_4^x - p_5^x)^2 + (p_4^y - p_5^y)^2 \end{cases}$$

We can isolate variable p^x in the first equation and then substitute p^x in the second equation which yields a system of equations $A_1 p^x = B_1 p^y + C_1$ and $A_2(p^y)^2 + B_2 p^y + C_2 = 0$, with $A_1 \neq 0$, $A_2 \neq 0$, B_1, B_2, C_1 and C_2 polynomials in the variables $p_1^x, p_1^y, \ldots, p_5^x, p_5^y$. The solution p^y then equals $-B_2/2A_2$ when $B_2^2 - 4A_2C_2 = 0$, or $(-B_2 \pm \sqrt{B_2^2 - 4A_2C_2})/2A_2$ when $B_2^2 - 4A_2C_2 > 0$. When $B_2^2 - 4A_2C_2 < 0$, no real solution p^y exists. Let D be the finite domain containing the active domain of R. Then, all computations involved can be performed with product variables ranging over D. Hence, by Proposition 5.25, there exists an SPFOL$^{\mathbf{Q}}$ program that, upon input R, has the desired point-set as output.

- *The geometric predicate* `c-c-crossing`$(\vec{p}, \vec{p}_1, \ldots, \vec{p}_6)$. First, we observe that the Boolean conditions "$\vec{p}_2$ and $\vec{p}_3$ are distinct," and "$\vec{p}_5$ and $\vec{p}_6$ are distinct," with $\vec{p}_2, \vec{p}_3, \vec{p}_5, \vec{p}_6$ points of R, can be expressed in FO + poly, whence, by Proposition 5.23, they can be also expressed in SPFOL$^{\mathbf{Q}}$. Next, the intersection point $\vec{p}$ has to satisfy the equations

$$\begin{cases} (p^x - p_1^x)^2 + (p^y - p_1^y)^2 = (p_2^x - p_3^x)^2 + (p_2^y - p_3^y)^2 & (1) \\ (p^x - p_4^x)^2 + (p^y - p_4^y)^2 = (p_5^x - p_6^x)^2 + (p_5^y - p_6^y)^2 & (2) \end{cases}$$

which is equivalent with the system consisting of equation (1), and equation (2) minus equation (1). The latter equation is of the form $Ap^x + Bp^y + C = 0$, with $A \neq 0$, B, and C polynomials in the variables $p_1^x, p_1^y, \ldots, p_6^x, p_6^y$. Hence, the initial problem is reduced to the computation of the intersection of a line and a circle, both defined by points of which the coordinates are described as polynomials in variables that range over D, where D is the finite domain containing the active domain of R. This problem is solved above. ■

We are now ready to prove the following result:

Proposition 5.40 *Every* SafeEuql*-expressible query can be expressed in* SPFOL$^{\mathbf{Q}}$.

Proof. Let Q be a query which is expressible in SafeEuql by the formula φ.

First, we observe that, for each point variable in a SafeEuql formula, only a finite number of points can be substituted to satisfy that formula. We show how the finite superset D, which contains all coordinates of these points, can be computed in SPFOL$^{\mathbf{Q}}$ by induction on the structure of the SafeEuql formula φ.

We start with D equal to $\{0, 1\}$.

For an atomic SafeEuql formula $\hat{R}(\vec{p}_1, \ldots, \vec{p}_k)$, we augment the domain D with the active domain of the input database.

Assume now SafeEuql formulae $\psi_1(\vec{p}_1, \ldots, \vec{p}_k)$ and $\psi_2(\vec{q}_1, \ldots, \vec{q}_l)$. Let D_1 and D_2 be the finite domains which contain the coordinates of the finitely many points that can be substituted for point variables of, respectively, ψ_1 and ψ_2, to satisfy, respectively, ψ_1 and ψ_2.

Consider a subformula of φ of the form

$$\psi_1(\vec{p}_1, \ldots, \vec{p}_k) \wedge \texttt{geom-pred}(\vec{p}, \vec{p}_{i_1}, \ldots, \vec{p}_{i_6}),$$

with $1 \leq i_1, \ldots, i_6 \leq k$ and `geom-pred` one of the following geometric predicates `l-l-crossing`, `l-c-crossing`, or `c-c-crossing`. Then, let D be equal to $D_1 \cup D_*$ where D_* is the finite domain of all coordinates of points in the output of the SPFOL$^{\mathbf{Q}}$ program of Lemma 5.39 for the corresponding geometric predicate $\texttt{geom-pred}(\vec{p}, \vec{p}_{i_1}, \ldots, \vec{p}_{i_6})$ upon input D_1.

For subformulae of φ of the following form

1. $\psi_1(\vec{p}_1, \ldots, \vec{p}_k) \wedge \psi_2(\vec{q}_1, \ldots, \vec{q}_l)$
2. $\psi_1(\vec{p}_1, \ldots, \vec{p}_k) \vee \psi_2(\vec{p}_1, \ldots, \vec{p}_k)$ or,
3. $\psi_1(\vec{p}_1, \ldots, \vec{p}_k) \wedge \neg\psi_2(\vec{p}_{i_1}, \ldots, \vec{p}_{i_m})$,

with $1 \leq i_1, \ldots, i_m \leq k$, let D be the union of D_1 and D_2.

In all other cases, D remains unchanged.

Clearly, there exists an SPFOL$^{\mathbf{Q}}$ program which outputs D upon input a possible input of Q. Moreover, this SPFOL$^{\mathbf{Q}}$ program expresses an active range superset query of Q. Furthermore, every SafeEuql formula has an equivalent FO + poly

formula [63], whence Q can also be expressed in FO + poly. Then, by Lemma 5.22, Q can be expressed in SPFOL$^{\mathbf{Q}}$. ■

In the following, we say that a SafeEuql formula $\varphi(\vec{p})$ represents a finite domain D if the sentence $\{(x, 0) \mid (D(x)\} = \{\vec{p} \mid \varphi(\vec{p})\}$.

To prove the converse of Proposition 5.40, we use the following lemma.

Lemma 5.41 *Let Q_1 and Q_2 be expressions built from the variables $p_1, \ldots, p_m$, and the rational numbers, using addition, multiplication, and absolute-value square rooting. There exists a* SafeEuql *formula φ such that*

$$\begin{aligned}\{(\vec{p}, \vec{p}_1, \ldots, \vec{p}_m) \mid \varphi(\vec{p}, \vec{p}_1, \ldots, \vec{p}_m)\} = \\ \{((p,0),(p_1,0),\ldots,(p_m,0)) \mid D(p_1) \wedge \ldots \wedge D(p_m) \\ \wedge Q_1(p_1,\ldots,p_m)p = Q_2(p_1,\ldots,p_m) \wedge Q_1(p_1,\ldots,p_m) \neq 0\},\end{aligned}$$

provided that a representation of D, the domain of the variables $p_1, \ldots, p_m$, can be computed from the input in SafeEuql.

Proof. First, we observe that the point $(p, 0)$, with p a rational number, can easily be constructed with ruler and compass from the points $\vec{0}$ and $\vec{e}_1$. Second, we show that addition, multiplication, and absolute-value square rooting on variables bound by D can be computed by ruler and compass. Thereto, consider the geometric constructions for multiplication, shown in Figure 3.6, addition and absolute-value square rooting, shown in Figure 5.1. All these geometric constructions can be performed in SafeEuql, assuming that we add conjuncts that state that the coordinates of $\vec{p}$ and $\vec{q}$ satisfy the SafeEuql formula representing D. By a simple induction on the structure of the expressions under consideration, we obtain a SafeEuql formula φ with the desired result. ■

We are now ready to prove the converse of Proposition 5.40.

Proposition 5.42 *Every* SPFOL$^{\mathbf{Q}}$*-expressible query can be expressed in* SafeEuql.

Proof. Since an SPFOL$^{\mathbf{Q}}$ program is defined by safe FO + linear + P$(D_1, \ldots, D_k)$ formulae (in which the polynomials have rational coefficients), it suffices to show that we can translate an arbitrary safe FO + linear + P$(D_1, \ldots, D_k)$ formula into a SafeEuql formula, given SafeEuql formulae that represent the finite domains $D_1, \ldots, D_k$. To do the translation, we observe that, in the SafeEuql query to be constructed, the point variables that occur in input relational predicates correspond to pairs of product variables in the input relational predicates of the given SPFOL$^{\mathbf{Q}}$

Figure 5.1: Geometric construction for addition (left), and absolute-value square rooting (right).

program. However, it is much easier to represent a product variable p by a point variable $\vec{p}$ corresponding to the point $(p, 0)$. This transition is possible in SafeEuql, as the points $\vec{p} = (p, 0)$ and $\vec{q} = (q, 0)$ can be constructed from the point $\vec{r} = (p, q)$ with ruler and compass. This technique can also be used to construct a SafeEuql formula which represents the active domain of the input database. To obtain the output point variables of the SafeEuql program, which correspond to pairs of output variables of the SPFOL$^{\mathbf{Q}}$ program, the reverse construction is required, which can also be done with ruler and compass.

Assume now finite domains $D_1, \ldots, D_k$ for which there exists SafeEuql formulae $\varphi_1, \ldots, \varphi_k$ such that $\{\vec{p} \mid \varphi_i(\vec{p})\}$ equals $\{(x, 0) \mid D_i(x)\}$, with $1 \leq i \leq k$. Let $\varphi(x_1, \ldots, x_{2n})$ be a safe FO + linear + P$(D_1, \ldots, D_k)$ formula. The SafeEuql formula is obtained by inductively translating φ to SafeEuql syntax.

Every occurrence of an atomic formula $R(t_1, \ldots, t_{2k})$ or $\texttt{adom}(t)$, with t real or product variables, is replaced by a SafeEuql formula as explained above.

Every occurrence of an atomic formula $Q(p_1, \ldots, p_m) > 0$, with Q an expression built from the product variables $p_1, \ldots, p_m$ and the rational numbers, using addition, multiplication and absolute-value square rooting, is replaced by

$$\varphi_Q(\vec{p}, \vec{p}_1, \ldots, \vec{p}_m) \wedge \texttt{on-same-side}(\vec{p}, \vec{e}_1, \vec{0}, \vec{e}_2),$$

with φ_Q the SafeEuql formula of Lemma 5.41 applied to $Q_1 = 1$ and $Q_2 = Q$. Similar formulae can be constructed for the formulae $Q(p_1, \ldots, p_m)\theta 0$ with $\theta \in \{\geq, <, \leq, =, \neq\}$. Notice that product variables in φ are appropriately bound to finite domains, and, hence, during the translation, we can add, for each point variable

representing a product variable, a conjunct stating that the point variable belongs to the appropriate domain.

Every occurrence of an atomic formula

$$Q_1(p_1, \ldots, p_m)x = Q_2(p_1, \ldots, p_m) \wedge Q_1(p_1, \ldots, p_m) \neq 0,$$

with Q_1 and Q_2 expressions built from the product variables $p_1, \ldots, p_m$ and the rational numbers, using addition, multiplication and absolute-value square rooting, is replaced by the SafeEuql formula $\varphi_Q(\vec{p}, \vec{p}_1, \ldots, \vec{p}_m)$, where $\varphi_Q(\vec{p}, \vec{p}_1, \ldots, \vec{p}_m)$ is the result of Lemma 5.41 applied to Q_1 and Q_2. As explained above, we can add, for each point variable representing a product variable, a conjuct stating that the point variable belongs to the domain to which the product variable is bound.

This concludes the basis of the induction. The rest of the induction process is straightforward. ■

The combination of Proposition 5.40 and Proposition 5.42 yields the main result of this subsection:

Theorem 5.43 *The languages* SPFOL$^{\mathbf{Q}}$ *and* SafeEuql *are equivalent in expressive power.*

5.3.4 Expressiveness of PFOL

We now summarize our results and show that PFOL can express a wide range of natural, **A**-linear FO + poly-expressible queries.

Definition 5.44 An FO + poly^{lin} query Q is *constructible* if there exists an SPFOL program that computes, from the canonical finite representation of the input, the active range of the canonical finite representation of the output, or, equivalent, an active range superset of the finite representation of the output. ■

We claim that the notion of constructibility is a natural notion, which, to a certain extent, can be seen as a generalization of the notion of domain preservation in the relational model to the context of the linear constraint database model. Indeed, from the equivalence of SPFOL and SafeEuql (Theorem 5.43), one can argue that constructibility implies that the output can be "assembled" from "material" "constructed" from the "building blocks" of the input; in the relational model, domain preservation means that the output can be "assembled" from the "building blocks," i.e., the entries, of the input.

We give an example of a constructible query.

Example 5.45 Let S be an arbitrary semi-linear set. Clearly, a bounding box for S is also a bounding box for both the convex closure and the affine support of S. It is now easy to see that a superset of the active domain of the canonical finite representation of the convex closure, respectively, the affine support, of S can be computed from the canonical finite representation of S. As both the convex-closure query and the affine-support query on arbitrary semi-linear sets are FO + poly-expressible, we may conclude that they are also constructible. ■

We can now state the following result:

Theorem 5.46 *The class of queries expressible in* PFOL *is the same as the class of constructible queries.*

Proof. Clearly, by Corollary 5.35 and Lemma 5.24, every query expressible in PFOL is also a constructible query.

Conversely, for each constructible query Q, consider the query $\overline{Q}$ which computes, upon as input the canonical finite representation of the input of Q, as output the canonical finite representation of the output of Q. By Lemmas 5.22 and 5.31, $\overline{Q}$ is expressible in SPFOL, whence certainly in PFOL. Since the encoding and decoding algorithms, computing the canonical finite representation of semi-linear set, respectively re-computing a semi-linear set from any of its finite representations, are expressible in PFOL, there exists a PFOL program which computes Q. ■

The relevance of Theorem 5.46 with respect to the expressive power of PFOL, is that PFOL can compute a wide range of natural, **A**-linear queries, while remaining sound for the **A**-linear constraint databases. The observation that SPFOL is closely related to SafeEuql, a geometrically inspired point-based language, suggests that PFOL, while coordinate-based just like FO + linear, is geometrically much more meaningful than the latter.

We conclude this section with showing that *not* all FO + poly^{lin} queries are constructible queries, whence PFOL is not equivalent to FO + poly^{lin}, confirming what one might have intuitively expected.

Let $\vec{p}$ and $\vec{q}$ be arbitrary points of the plane such that $\vec{0}$, $\vec{p}$, and $\vec{q}$ are not collinear. It is well-known that the point $\vec{r}$ on the line segment defined by $\vec{p}$ and $\vec{q}$ such that the angle defined by $\vec{0}$, $\vec{p}$, and $\vec{r}$ is the trisection of the angle defined by $\vec{0}$, $\vec{p}$, and $\vec{q}$ cannot be constructed with ruler and compass from $\vec{0}$, $\vec{e}_1$, $\vec{p}$, and $\vec{q}$. This result remains true even if to these points a fixed (i.e., independent of $\vec{p}$ and $\vec{q}$) finite set of points is added [59]. By the equivalence between SPFOL$^{\mathbf{Q}}$ and SafeEuql, established in Theorem 5.43, and by Theorem 5.46, we can now state the following result.

Proposition 5.47 *The* trisection *query of type* $[0,2] \to [0,2]$ *computing for each point* $\vec{p} \neq \vec{0}$ *in the input all points on a trisector of the angle defined by* $\vec{0}$, $\vec{e}_1$, *and* $\vec{p}$ *cannot be expressed in* PFOL.

Proof. Assume to the contrary that there exists a PFOL program that computes the trisection query. We now consider the query which outputs, on finite input, the output of the trisection query on that input intersected with the line $x = 1$, and, on infinite input, the empty set. Clearly, if the trisection query is expressible in PFOL, this modified trisection query is also expressible in PFOL. Moreover, the latter query yields finite outputs upon finite inputs, whence it is in PFOL-finite. By Theoreom 5.34, there exists an SPFOL program

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, x_2) \mid \varphi(R; x_1, x_2)\},$$

computing that query.

If all expressions in φ have rational coefficients, there exists, by Theorem 5.43, an equivalent SafeEuql expression, whence the trisection of an angle defined by the points $\vec{0}$, $\vec{e}_1$, and an arbitrary point $\vec{p}$ can be constructed by ruler and compass, which yields a contradiction.

Thus, assume that some expressions in φ have real algebraic coefficients. Denote by C_a the finite set of real algebraic coefficients occuring in a polynomial of φ and let $C_{\vec{a}}$ be the set of points $\{(a, 0) \mid a \in C\}$. We can then generalize Lemma 5.41 to expressions with coefficients from C_a and SafeEuql expressions with input $C_{\vec{a}} \cup \{\vec{p}\}$. Hence, using a similar proof as that of Theorem 5.42, we can construct a SafeEuql expression $\psi(R, S; \vec{x})$ such that, for every finite relation R,

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, x_2) \mid \varphi(R; x_1, x_2)\},$$

and $\psi(R, C_{\vec{a}}; \vec{x})$ compute the same set of points in the plane. Hence, we can construct the trisection of an angle defined by the points $\vec{0}$, $\vec{e}_1$, and an arbitrary point $\vec{p}$ by ruler and compass from the finite set of points $C_{\vec{a}} \cup \{\vec{0}, \vec{e}_1, \vec{p}\}$, a contradiction. ∎

5.4 Expressiveness of FO + linear Extended with Operators

We now turn our attention to the expressive power of the query language FO + linear + $\mathcal{O}$ proposed in Section 5.1. We present here an extension of FO + **A**-linear with **A**-linear operators of which we prove that it has the same expressive power as PFOL.

Definition 5.48
- The linear operator `line`[12] of type $[0,2] \to [0,2]$ is defined by the FO + poly formula $(\exists u)(\exists v)(R(u,v) \wedge uy = vx)$. In words, the operator `line` returns the union of all lines through some point of the input and the origin $(0,0)$.[13]
- The **A**-linear operator `sqrt`[12] of type $[0,1] \to [0,1]$ is defined by the FO + poly formula $(\exists u)(R(u) \wedge ((u \geq 0 \wedge x^2 = u) \vee (u < 0 \wedge -x^2 = u))$. In words, the operator `sqrt` returns those real numbers that are the absolute-value square root of a real number in the input. ■

We abbreviate the extension of FO + **A**-linear with the operators `line` and `sqrt` to FO + linear + $\{\text{line}, \text{sqrt}\}$. We now prove that FO + linear + $\{\text{line}, \text{sqrt}\}$ has the same expressive power as PFOL. (For this purpose, we only consider the geometric part of FO + linear + $\{\text{line}, \text{sqrt}\}$.)

To do so, we need the following lemmas.

Lemma 5.49 *Let R be of type $[0,1]$. The predicate* `product`$(R; p, x, y)$ *which evaluates to* true *if R is finite, p is in R, and $y = px$ can be expressed in* FO+linear+$\{\text{line}\}$ *(whence in* FO + linear + $\{\text{line}, \text{sqrt}\}$*).*

Proof. We can check in FO + linear, whence in FO + linear + $\{\text{line}\}$, whether R is finite. If R is finite, $p = 0$, and p is in R, then `product`$(R; p, x, y)$ evaluates to *true* if and only if $y = 0$. It remains to explain what to do if R is finite, $p \neq 0$, and p is in R.

Let $L =$`line`$(\{(x,y) \mid x = 1 \wedge R(y) \wedge y \neq 0\})$. Since the operand is finite and does not contain the origin of the plane, L is a finite union of lines. By Proposition 3.30, we can select in FO + **A**-linear, whence in FO + linear + $\{\text{line}\}$, the line L_p going through the point $(1,p)$. Since this line also goes through the point $(0,0)$, it readily follows that `product`$(R; p, x, y)$ is *true* if and only if the point (x,y) is on L_p. ■

Lemma 5.50 *Every* PFOL*-expressible query can be expressed in* FO + linear + $\{\text{line}, \text{sqrt}\}$.

Proof. Consider the PFOL program

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x);$$
$$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\},$$

[12]From their definitions, it is not obvious that `line` and `sqrt` are **A**-linear operators. However, since PFOL can only express **A**-linear queries (Theorem 5.1), the proof of Lemma 5.51 also entails a proof of the linearity of these operators.

[13]Notice that, if $R(0,0)$ is *true*, then the operator `line` returns the entire plane.

with, for $i = 1, \ldots, k$, D_i domain symbols, φ_i an FO + linear + P$(D_1, \ldots, D_{i-1})$ formula, and φ an FO + linear + P$(D_1, \ldots, D_k)$ formula.

First, we show how to obtain FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions which compute the domains $D_1, \ldots, D_k$ according to the semantics of PFOL. The FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions are obtained by inductively translating the expressions $\varphi_1, \ldots, \varphi_k$ to FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_k$.

For the basis of this induction, we observe that φ_1 is an FO + **A**-linear expression, whence also an FO + linear + $\{\text{line}, \text{sqrt}\}$ expression. Let $\tilde{\varphi}_1$ be the formula $\texttt{finite}(\{(x) \mid \varphi_1(x)\}) \wedge \varphi_1$, with `finite` the FO + **A**-linear query which decides whether a **A**-semi-linear set is finite. Clearly, $\tilde{\varphi}_1$, evaluated in the standard manner, yields the correct interpretation of D_1, independent of whether $\{(x) \mid \varphi_1(x)\}$ is finite or not.

Now, assume that, for $1 \leq i < k$, there are FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_i$ computing the domains $D_1, \ldots, D_i$ according to the semantics of PFOL. Consider the FO + linear + P$(D_1, \ldots, D_i)$ expression φ_{i+1}. In φ_{i+1}, we substitute every subformula of the form $(\exists p \in D_j)\psi$, $1 \leq j \leq i$, by $(\exists p)(\tilde{\varphi}_j(p) \wedge \tilde{\psi})$, where $\tilde{\psi}$ is the formula obtained from ψ as follows:

- every occurrence of an atomic formula of the form $pt_1 = t_2$ is replaced by the FO + linear + $\{\text{line}, \text{sqrt}\}$ formula simulating $\texttt{product}(\{(x) \mid \tilde{\varphi}_j(x)\}; p, t_1, t_2)$ (Lemma 5.49); and
- every occurrence of an atomic formula of the form $t = \sqrt{|p|}$ is replaced by the FO + linear + $\{\text{line}, \text{sqrt}\}$ formula simulating

$$(p \geq 0 \wedge \texttt{product}(\{(x) \mid \texttt{sqrt}(\{(x) \mid \tilde{\varphi}_j(x)\})(x)\}; t, t, p)) \vee \\ (p < 0 \wedge \texttt{product}(\{(x) \mid \texttt{sqrt}(\{(x) \mid \tilde{\varphi}_j(x)\})(x)\}; t, t, -p)).$$

We abbreviate the FO + linear + $\{\text{line}, \text{sqrt}\}$ formula obtained in this way as $\hat{\varphi}_{i+1}$. By Lemma 5.49, $\hat{\varphi}_{i+1}$ and φ_{i+1} are equivalent when evaluated in the standard manner. Therefore, the FO + linear + $\{\text{line}, \text{sqrt}\}$ expression $\tilde{\varphi}_{i+1} = \texttt{finite}(\{(x) \mid \hat{\varphi}_{i+1}(x)\}) \wedge \hat{\varphi}_{i+1}(x)$, evaluated in the standard manner, yields the correct interpretation of D_{i+1}, independent of whether $\{(x) \mid \varphi_{i+1}(x)\}$ is finite or not.

Thus, all domains can be computed in FO + linear + $\{\text{line}, \text{sqrt}\}$.

The only thing that remains is to show that the FO + linear + P$(D_1, \ldots, D_k)$ expression φ can be translated into an FO + linear + $\{\text{line}, \text{sqrt}\}$ expression, for which, of course, the same technique applies: every subformula $(\exists p \in D_j)\psi$, $1 \leq j \leq k$, is replaced by an FO + linear + $\{\text{line}, \text{sqrt}\}$ formula $(\exists p)(\tilde{\varphi}_j(p) \wedge \tilde{\psi})$ in the way explained above. The resulting FO + linear + $\{\text{line}, \text{sqrt}\}$ formula clearly expresses the same query as the original PFOL program. ∎

Lemma 5.51 *Every* FO + linear + {line, sqrt}*-expressible query can be expressed in* PFOL*.*

Proof. It suffices to show that both `line` and `sqrt` can be simulated by a PFOL program to establish the result. The main difficulty in doing so is that, in FO + linear + {line, sqrt}, both `line` and `sqrt` can take an infinite input. However, it turns out to be always possible to select a finite number of points from the input such that the output can be constructed from the result of the operator applied to this finite set of points and the general structure of the input.

Since the operator `line` takes a two-dimensional set as input, whereas the operator `sqrt` takes a one-dimensional set as input, we start with the latter, simpler, case.

Thus, let R be of type $[0, 1]$. As observed above, the naive "solution" of using the PFOL program

$$D_1 \leftarrow R(x);$$
$$\{(x) \mid (\exists p \in D_1)(x = \sqrt{|p|})\}$$

fails, since R may be infinite. Since $\{x|\texttt{sqrt}(R; x)\}$ equals $\{x|\texttt{sqrt}(\{x|R(x) \wedge x \geq 0\}; x) \vee \texttt{sqrt}(\{x|R(-x) \wedge x \geq 0\}, x)\}$, we assume in the following that R contains only positive real numbers. A semi-algebraic set, in particular also a semi-linear set, has only a finite number of connected components, whence R is either empty, or a line, or a finite union of isolated points, intervals, and half-lines. If we apply the `sqrt` to the isolated points and the end points of intervals and half lines in R, the output can be obtained from these by "filling in" the intervals and half lines appropriately. Thus, let D_1 be the boundary of R, which consists of all isolated points and end points of line segments and half-lines of R. Clearly, D_1 is a finite set which can be computed in FO + linear (Theorem 3.15). The following FO + linear + P(D_1) expression then yields `sqrt`(R):

$$\begin{aligned}\{(x) \mid\ & (\neg(\exists y)R(y) \Rightarrow \texttt{false}) \vee \\ & (\neg(\exists y)D_1(y) \Rightarrow R(x)) \vee \\ & (\exists p \in D_1)(R(p) \wedge x = \sqrt{|p|}) \vee \\ & (\exists p_1 \in D_1)(\exists p_2 \in D_1)(\exists y_1)(\exists y_2)((\forall z)(p_1 < z < p_2 \Rightarrow R(z)) \wedge \\ & \qquad\qquad y_1 = \sqrt{|p_1|} \wedge y_2 = \sqrt{|p_2|} \wedge y_1 < x < y_2) \vee \\ & (\exists p \in D_1)(\exists y)((\forall z)(p < z \Rightarrow R(z)) \wedge y = \sqrt{|p|} \wedge y < x) \vee \\ & (\exists p \in D_1)(\exists y)((\forall z)(p > z \Rightarrow R(z)) \wedge y = \sqrt{|p|} \wedge y > x)\}.\end{aligned}$$

We now turn to the operator `line`, which takes a (possibly infinite) two-dimensional semi-linear set as input. Let R, of type $[0, 2]$, be the input predicate. Rather than providing the PFOL program, we explain how to construct it.

Since `line`(R) yields the entire plane if R contains the origin $(0, 0)$, we can deal with this case separately and assume for the remainder of the proof that R does *not*

contain the origin. We furthermore assume that R is *bounded* and explain afterwards which changes must be made to the reasoning below if R is not necessarily bounded.

Again since a semi-linear set has only a finite number of connected components, R is a finite union of isolated points, intervals, and polygons. We first compute the isolated points, the end points of the intervals, and the corner points of the polygons, which together are the *special points* of R, with Algorithm 4.33. Let S be the finite set of special points of R, augmented with the points $(1, 0)$ and $(0, 1)$[14], and let D_1 be the finite set consisting of the coordinates of points of S. Let L be the union of all the lines through the origin and a point of S. Obviously, L can be computed by the PFOL expression

$$\{(x, y) \mid (\exists p \in D_1)(\exists q \in D_1)(S(p, q) \wedge py = qx)\}.$$

The lines of L induce a partition of the plane, consisting of the origin (which always belongs to the output of $\texttt{line}(R)$, except when R is empty), the open half-lines in which the origin divides the lines of L, and the open convex angular sectors between successive half-lines. The output of $\texttt{line}(R)$ consists of the union of all the one- and two-dimensional classes of this partition which have a non-empty intersection with R and their mirror images with respect to the origin, augmented with the origin if R is not empty.

The union of the open half-lines which have a non-empty intersection with R can be computed by the PFOL expression

$$\{(x, y) \mid (\exists p \in D_1)(\exists p \in D_2)(\exists \lambda)(\exists \mu)(\exists z_1)(\exists z_2)(S(p, q) \wedge \\ \lambda > 0 \wedge \mu > 0 \wedge R(z_1, z_2) \wedge z_1 = \lambda p \wedge z_2 = \lambda q \wedge x = \mu p \wedge y = \mu q\}.$$

The open convex angular sector defined by two *non-collinear* open half-lines consists of all mid-points of a point of the first and a point of the second half-line.[15] Such an angular sector is a class of the partition if the defining half-lines are classes of the partition, and if no other half-line which is a class of the partition has a point in common with the angular sector. Notice that, by the addition of the points $(1, 0)$ and $(0, 1)$ to S, the partition is guaranteed not to contain angular sectors of $180°$, whence the above observations can be used to construct a PFOL expression computing the union of the open convex angular sectors which have a non-empty intersection with R.

Finally, to conclude the case where R is bounded, let U be the union of all one- and two-dimensional classes of the partition which have a non-empty intersection with

[14]The reason for the addition of these two points will be explained a little later.

[15]Notice that if we would have worked with lines rather than half-lines, this part of the proof would have failed; the same construction applied to full lines does not yield the union of the relevant angular sector and its mirror image with respect to the origin, but the entire plane.

R. Then the output of $\texttt{line}(R)$ is computed by the PFOL expression

$$\{(x,y) \mid (x = 0 \wedge y = 0 \wedge (\exists u)(\exists v)R(u,v)) \vee U(x,y) \vee \\ (\exists z_1)(\exists z_2)(U(z_1,z_2) \wedge x + z_1 = 0 \wedge y + z_2 = 0)\}.$$

As shown in Subsection 5.3.1, the above reasoning can be generalized to unbounded semi-linear sets by considering "special points at infinity," which are represented by pairs of directional numbers. (Conceptually, all semi-linear sets can then be treated as if they were bounded.) It was also shown in that subsection that all the required constructions can be simulated in PFOL. This completes the proof. ∎

Lemmas 5.51 and 5.50 together yield the following conclusion.

Theorem 5.52 *The linear constraint query languages* PFOL *and* FO + **A**-linear *extended with the operators* `line` *and* `sqrt` *have the same expressive power.*

By examining the proofs of the previous lemmas, it also follows that the restriction of PFOL in which square-root terms are disallowed is equivalent to FO+linear+{line}. Alternatively, one may extend PFOL by adding cube-root terms, etc. One can easily see that the above proofs generalize provided FO + linear + {line, sqrt} is extended by operators corresponding to the added terms.

Of course, the above result does not settle the expressiveness of extensions of FO + linear with operators in general. In particular, the questions whether there exists an extension of FO + linear with (a recursively enumerable set of) operators which captures precisely the FO + poly^{lin} queries remains open.

5.5 Query Languages Complete for FO + poly$^{\mathbf{lin}}$

In this section, we are concerned with query languages that are sound and complete for the FO + poly$^{\mathbf{z}\text{-}lin}$ queries on the one hand and the FO + poly$^{\mathbf{a}\text{-}lin}$ queries on the other hand. The most straightforward way to obtain such a query language is to discover an algorithm to decide whether an FO + poly formula induces a **Z**-linear query or an **A**-linear query. Unfortunately, such an algorithm does not exist, neither in the **Z**-linear, nor in the **A**-linear setting, because we can again invoke Theorem 3.38, this time with $\mathcal{C}_1 = \mathcal{C}_2$ the class of all polynomial constraint relations and $\mathcal{E}$ the property of inducing a **Z**-linear or **A**-linear query, yielding

Theorem 5.53 *It is undecidable whether an* FO + poly *formula induces a* **Z**-*linear or* **A**-*linear query.*

Observe that Theorem 5.53 does not rule out that one can isolate a subset of the FO + poly formulae which expresses precisely the FO + poly$^{\mathbf{Z}\text{-}lin}$ queries or the FO + poly$^{\mathbf{a}\text{-}lin}$ queries, in the same way that the undecidability of domain independence in the relational calculus is not in contradiction with the existence of a sublanguage of the relational calculus which precisely expresses the domain-independent relational calculus queries [72].

In the remainder of this section, we propose query languages which are sound and complete for the FO + poly$^{\mathbf{Z}\text{-}lin}$ queries, respectively the FO + poly$^{\mathbf{a}\text{-}lin}$ queries.

First, we use the decidability results on semi-linearity of semi-algebraic sets, elaborated upon in Chapter 4, to show that there exist query languages which are sound and complete for the FO + poly$^{\mathbf{Z}\text{-}lin}$ queries, respectively the FO + poly$^{\mathbf{a}\text{-}lin}$ queries. We recall from Chapter 4 that it can be decided whether a semi-algebraic set is **Z**-semi-linear or **A**-semi-linear. We start with a query language that is complete for the FO + poly$^{\mathbf{Z}\text{-}lin}$ queries. Syntactically, the query language concerned is just FO + poly. Semantically, the standard output of an FO + poly query is replaced by the empty set if it is not a **Z**-semi-linear set. The language thus obtained is obviously sound and complete (the standard output of a linear FO + poly query is not modified) for the FO + poly$^{\mathbf{Z}\text{-}lin}$ queries. This also works in the **A**-linear setting to obtain a query language sound and complete for the FO + poly$^{\mathbf{a}\text{-}lin}$ queries. In the **A**-linear case, however, there exists an FO + poly expression that decides whether a semi-algebraic set is **A**-semi-linear (Theorem 4.37). Hence, by substituting this FO + poly expression deciding **A**-semi-linearity in the above described manipulation of FO + poly formulae, we obtain a syntactically defined fragment of FO + poly which is sound and complete for the FO + poly$^{\mathbf{a}\text{-}lin}$ queries with respect to the standard semantics. As a consequence, there exists a recursively enumerable subset of FO + poly which expresses precisely the **A**-linear queries expressible in FO + poly. Observe that we can generalize the above described manipulation of FO + poly to arbitrary properties of semi-algebraic sets.

Proposition 5.54 *Let $\mathcal{E}$ be a property of semi-algebraic sets which can be expressed in* FO + poly *and let $\mathcal{C}$ be the class of semi-algebraic sets with property $\mathcal{E}$. There exists a syntactically defined fragment of* FO + poly *which is sound and complete for the* FO + poly *queries that return outputs in $\mathcal{C}$ upon inputs in $\mathcal{C}$.*

Admittedly, the languages thus obtained neither are particularly elegant nor have much practical value, but, at least, their existence justifies the search for more natural sound and complete languages for the FO + poly^{lin} queries.

Second, we consider the class of *polynomial-restricted queries*, exhibited by Benedikt and Libkin [10], which form a syntactically defined fragment of FO + poly of which these authors have shown it is sound and complete for the FO + poly-expressible

queries from finite **A**-semi-linear inputs to finite **A**-semi-linear outputs. Hence, by Theorem 5.15, if we let $\mathcal{P}$ be the **A**-linear queries expressible in FO + poly, and $\mathcal{Q}$ the polynomial-restricted queries, we obtain the following result.

Corollary 5.55 *The* **A***-linear query language Lift*($\mathcal{Q}$)*, with* $\mathcal{Q}$ *the class of polynomial-restricted* FO + poly *queries, is sound and complete for the* FO + poly$^{\mathbf{a}\text{-}lin}$ *queries.*

A simular approach to obtain a linear query language complete for the FO + poly$^{\mathbf{a}\text{-}lin}$ queries using Theorem 5.15, is possible. As the query deciding finiteness is expressible in FO + linear, whence in FO + poly, it follows from Proposition 5.54 that there exists a syntactically defined fragment of FO + poly which is sound and complete for the FO + poly queries from finite **A**-semi-linear sets to finite **A**-semi-linear sets. Obviously, applying Theorem 5.15 on this syntactically definable fragment of FO + poly yields a query language sound and complete for the FO + poly$^{\mathbf{a}\text{-}lin}$ queries.

Finally, we mention that it is possible to define a language complete for *all* computable linear queries, even those not expressible within FO + poly. It is shown in [45] that it is possible to augment FO + linear with a while-loop such that the resulting query language is sound and complete for the computable linear queries.

Chapter 6

Discussion

We conclude this dissertation with a short discussion on the implementation of the linear query languages introduced in Chapter 5. We also propose some directions for future research on linear query languages.

First, we summarize the main contributions of this dissertation.

6.1 Main Results

In this dissertation, we studied linear query languages for the linear constraint database model and their expressive power. Our main results are as follows.

First, we provided a list of queries which can be expressed in FO + linear. This list of queries turned out to be a useful instrument to show the expressibility in FO + linear of several other queries. We also provided a tool which can be used to prove non-expressibility of a query in FO + linear. We used that tool to prove that various fundamental linear queries cannot be expressed in FO + linear. We showed that the type of the coefficients of the linear constraints used in the linear constraint database model does influence certain expressibility results. We proved that it is undecidable whether a linear query expressible in FO + poly can be expressed in FO + linear. This results remains true for several subclasses of the FO + poly-expressible linear queries.

Second, we developed an algorithm, implementable in FO + poly, to compute the "special points" of a semi-linear set. These special points of a semi-linear set allow us

to obtain a decomposition of that semi-linear set into convex cells which is expressible in FO + poly. Moreover, we obtain algorithms, implementable in FO + poly, to compute a finite representation of an arbitrary semi-linear set and to recompute a semi-linear set from its finite representation.

Third, we showed that it is decidable in FO + poly whether a semi-algebraic set is **A**-semi-linear. We also showed that **Z**-semi-linearity of a semi-algebraic set is decidable, but not in FO + poly.

Fourth, we presented two sound extensions of FO + linear for linear queries expressible in FO+poly. One extension resulted in a query language of which the expressive power can be characterized in terms of the ruler and compass constructions in the two-dimensional plane. The other extension described how linear operators can be added to FO + linear in a sound way. As an example of the potential of this paradigm for extending FO + linear with linear operators, we presented two linear operators such that FO+linear extended with these two operators captures precisely all queries expressible in the query language obtained in the first extension. We also showed that there exist query languages which are complete for the linear queries expressible in FO + poly.

6.2 Some Remarks on Implementing the Query Languages PFOL and FO + linear + $\mathcal{O}$

The main purpose of this section is to argue that the two extensions of FO + linear introduced in Chapter 5 (FO + linear + $\mathcal{O}$ and PFOL) are not only query languages with a "theoretical" value, and, by doing so, to encourage the use of these linear query languages in implementations of the linear constraint database model. We exhibit in this section possible approaches to implement the query languages FO + linear + $\mathcal{O}$ and PFOL on top of a linear constraint database system (with FO + linear as linear query language). Some of the proposals, however, require further research to obtain a better insight in their feasibility.

We start this section with a brief discussion on the DEDALE system, which implements the linear constraint database model with FO + linear as linear query language.

6.2.1 On the Implementation of FO + linear

In this subsection, we briefly describe the DEDALE system, which is a first prototype of the linear constraint database model, recently developed by Grumbach et al. [31,

32]. The data considered in DEDALE consists of alphanumerical values and two-dimensional semi-linear sets. We concentrate here on the geometric part of the linear constraint database model and the query language FO + linear, as existing relational database systems can be used for the implementation of the alphanumerical part of the linear constraint database model and the query language FO + linear.

Because of the declarative style of FO + linear, queries formulated as expressions in FO + linear are less appropriate for query evaluation. By Theorem 2.13, the query language FO + linear is equivalent in expressive power to the linear constraint algebra, which has a more procedural style. Moreover, an FO + linear expression can be translated automatically into a linear constraint algebra expression. As formally described in Section 2.3, the operators of the linear constraint algebra, which work on semi-linear sets, consist of union, intersection, complement, Cartesian product, selection, and projection.

Roughly spoken, the DEDALE system implements the operators of the linear constraint algebra as follows[1]:

- projection is implemented using Fourier-Motzkin quantifier elimination on finite sets of linear inequalities;
- union, intersection, selection, and Cartesian product are performed by a symbolic manipulation on the constraints representing the semi-linear sets to which these operators are applied; the actual computation of these operators is postponed until an explicit call of the function which computes the polygon representation of a two-dimensional semi-linear set is made; and, finally,
- the complement of a two-dimensional semi-linear set is computed by making a cell decomposition of the plane using the lines defined by $a_1x_1 + a_2x_2 + a_3 = 0$, with $a_1x_1 + a_2x_2 + a_3$ a linear constraint term occuring in the linear constraint formula defining that semi-linear set.

[1] A program that can also be useful in this context, is the program called `cdd`, or the modern object-oriented version of it, `cdd+`. The program `cdd+` implements the Double Description Method of Motzkin et al. [29, 57] for generating all special points (corner points and points at infinity) of a general convex polyhedron in $\mathbf{R}^n$ given by a system of linear inequalities. The program can also be used for the reverse operation, i.e., computing the convex closure of a set of n-dimensional corner points and points at infinity. This means that one can move back and forth between an inequality representation and the finite representation of a n-dimensional polyhedron with `cdd+`. The program `cdd+` can be found on the Internet by following the link `http://www.ifor.math.ethz.ch/ifor/staff/fukuda/cdd_home/cdd.html`. Another example of an algorithm that computes the convex closure of a finite set of points in the n-dimensional space, is the *QuickHull* algorithm, developed by Barber et al. [7]. The program `qhull` implements the QuickHull algorithm, and can be found on the Internet by following the link `http://www.geom.umn.edu/locate/qhull`.

6.2.2 On the Implementation of FO + linear + $\mathcal{O}$

In this subsection, we describe how FO + linear + $\mathcal{O}$ can be implemented on top of a linear constraint database system with the linear constraint algebra as query language. Thereto, we extend the linear constraint algebra with the operators represented by $\mathcal{O}$ to obtain a more procedural query language which is equivalent to FO + linear + $\mathcal{O}$.

Let $\mathcal{O}$ be a set of operator names, each of which denotes a linear operator. For each operator name O in $\mathcal{O}$ of type $[m_1, n_1; \ldots ; m_k, n_k] \to [m, n]$, we add to the operators defining the linear constraint algebra, the operator

$$O(r_1, \ldots , r_k),$$

where $r_1, \ldots , r_k$ are syntactic linear constraint relations of type $[m_1, n_1], \ldots , [m_k, n_k]$ respectively. We call the resulting query language the linear constraint algebra + $\mathcal{O}$. We can now generalize the result of Proposition 2.13:

Proposition 6.1 *Every* FO + linear + $\mathcal{O}$ *expression can be converted effectively into a linear constraint algebra* + $\mathcal{O}$ *expression and vice-versa, in such a way that both express the same mapping from linear constraint database instances to linear constraint database instances, respectively at the semantic and syntactic level.*

Of course, the efficiency of the evaluation of expressions in the linear constraint algebra extended with $\mathcal{O}$-operators depends heavily on the availability of efficient algorithms for computing the operators represented by $\mathcal{O}$ and the efficiency of the evaluation of linear constraint algebra expressions. However, because of the severe limitations of FO + linear, one should at least consider extending FO + linear with a minimal set of aggregate operators to compute, for instance, distance, area, and volume. As a matter of fact, the DEDALE prototype system includes, besides the linear algebra operators, the following efficiently implementable linear operators:

- $\sigma_{\texttt{dist}\,\theta\,d}$ of type $[0, n; 0, n] \to [0, 0]$, which evaluates to *true* if there exists some point $\vec{p}$ in the first input relation and some point $\vec{q}$ in the second input relation such that $\texttt{distance}(\vec{p}, \vec{q})\,\theta\,d$ is satisfied;
- `axis` of type $[0, 2] \to [0, 2]$, which yields as output the affine support of the input relation, and;
- `median` of type $[0, n; 0, n; 0, n]$, which yields as output the empty set if one of the input relations consist of more than one point, and otherwise, the Voronoi-cell of the point in the third input relation with respect to the points in the first and in the second input relation.

6.2.3 On the Implementation of PFOL

The approach we follow here is based on the result established in Theorem 5.52 in which we showed that PFOL and FO + linear + $\{\text{line}, \text{sqrt}\}$ are equivalent query languages. For convenience, we reiterate the definition of the linear operators `line` and `sqrt`:

- The linear operator `line` of type $[0,2] \to [0,2]$ returns the union of all lines through some point of the input and the origin $(0,0)$.[2]
- The linear operator `sqrt` of type $[0,1] \to [0,1]$ returns those real numbers that are the square root of the absolute value of a real number in the input.

Combining Theorem 5.52 and Proposition 6.1, we immediately obtain the following result.

Proposition 6.2 *Every* PFOL *expression can be converted effectively into a linear constraint algebra expression with* `sqrt` *and* `line` *operators and vice-versa, in such a way that both express the same mapping from linear constraint database instances to linear constraint database instances, respectively at the semantic and syntactic level.*

For our purposes, we concentrate on the translation of a PFOL program into an FO + linear + $\{\text{line}, \text{sqrt}\}$ expression, which is done in the proof of Lemma 5.50. An inspection of the proof of Lemma 5.50 shows that this translation is indeed effective and, moreover, in the FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions obtained as the translation of PFOL programs, the operators `line` and `sqrt` are only applied to *finite* point-sets in two-dimensional and one-dimensional space, respectively. Obviously, the operators `line` and `sqrt` can be implemented efficiently on finite sets of points. Hence, we can conclude that the evaluation of FO + linear + $\{\text{line}, \text{sqrt}\}$ expressions in which the operators `line` and `sqrt` are applied to finite sets is as tractable as the evaluation of FO + linear expressions.

6.3 Directions for Future Research

We conclude this work with two possible directions for future research on linear query languages for the linear constraint database model.

[2]Notice that, if $R(0,0)$ is *true*, then the operator `line` returns the entire plane.

First, we must emphasize that we are still far away from a precise insight into the expressiveness of FO + linear on semi-linear figures. As a step towards answering this question, we could investigate the expressive power of FO + linear on certain subclasses of the semi-linear figures? For instance, in Section 3.2, we studied the expressibility in FO + linear of various geometric properties as "being parallel" and "having the same angle" on databases consisting of lines. Now, any database consisting of finite unions of non-zero-dimensional affine subspaces, can be encoded as a finite number of lines[3]. Indeed, we can generalize Proposition 3.30 to non-zero-dimensional affine subspaces, and Lemma 3.22 shows that the encoding of one affine subspace can be obtained as intersections of that affine subspace with the appropriate coordinate subspaces. Moreover, this encoding is expressible in FO + linear. The corresponding decoding is also expressible in FO + linear: given a set of lines, of which we can assume without loss of generality that these lines are pairwise non-parallel, the affine subspace spanned by these lines is obtained by computing the midpoint of every possible pair of points on these lines. Hence, similarly as in Theorem 5.15, we can characterize the expressive power of FO + linear on databases consisting of finite unions of affine subspaces in terms of the expressive power of FO + linear on databases consisting of finite unions of lines. A precise characterization of the expressiveness of FO + linear on finite unions of lines remains open, however.

Related to the above, it would be interesting to study subclasses of semi-linear figures for which it is decidable whether an FO + poly-expressible query which is sound with respect to a particular subclass of the semi-linear figures can be expressed in FO + linear. In Section 3.4, it is shown for various subclasses of the semi-linear figures that it is undecidable whether an FO + poly-expressible query which is sound with respect to one of these subclasses can be expressed in FO + linear. We might have more success, however, when the number of "components" of which the figures of a particular subclass consist is fixed, e.g., databases which consist of at most k affine subspaces, $k > 0$.

Second, in Chapter 5, we proposed extensions of FO + linear to capture richer subclasses of the FO + poly^{lin} queries. In our opinion, this direction deserves further exploration. The following are only illustrations of research questions one might ask:

- Can we find a finite set of operators such that FO + linear + $\mathcal{O}$ is complete for the linear queries?
- Can we find "practical" query languages which capture the linear queries expressible in FO + poly?

[3] A k-dimensional affine subspace can be encoded with $k - 1$ lines.

- Can we find query languages which are sound and complete for the linear queries that satisfy one of the various genericity conditions studied in [61]?
- What is the expressive power of PFOL augmented with all k-roots?

As a general conclusion of this chapter, we may say that there are still several exciting research directions concerning new languages, their expressiveness and their implementation that make future investigations of the linear constraint database model worthwhile.

Appendix A

Property SL and Semi-Linearity

The purpose of this appendix is giving a rigorous proof of the key property of Chapter 4, which states that an **R**-semi-algebraic set is **R**-semi-linear if and only if it satisfies Property SL. The proof of this claim is due to Freddy Dumortier [20].

For convenience, we reiterate Property SL:

Definition 4.1 Let $S \subseteq \mathbf{R}^n$. We say that S has Property SL if, for every point $\vec{p}$ of $\overline{S}$, there exists a neighborhood V of $\vec{p}$ such that, for every point $\vec{q}$ of V,

1. if $\vec{q}$ is in S, then $]\vec{p}, \vec{q}[$ is fully contained within S; and
2. if $\vec{q}$ is not in S, then $]\vec{p}, \vec{q}[$ is disjoint from S. ■

First, we observe that having Property SL is preserved under several operations.

Lemma A.1 *Having* Property SL *is preserved under set difference, complementation, finite intersection, and finite union.*

Proof.

1. *Difference*: Let $S_1, S_2 \subseteq \mathbf{R}^n$ both have Property SL. Let $\vec{p}$ be in $\overline{S_1 - S_2}$. Obviously, $\vec{p}$ is then also in $\overline{S_1}$. We now distinguish two cases.

   First, if $\vec{p}$ is in $\overline{S_2}$, then we choose a neighborhood V of $\vec{p}$ such that Definition 4.1 is satisfied with respect to both S_1 and S_2. A straightforward

verification then suffices to see that Definition 4.1 is also satisfied with respect to $S_1 - S_2$.

Second, if $\vec{p}$ is not in $\overline{S_2}$, then we choose a neighborhood V of $\vec{p}$ such that Definition 4.1 is satisfied with respect to S_1 and such that $V \cap \overline{S_2} = \emptyset$. Again, a straightforward verification then suffices to see that Definition 4.1 is also satisfied with respect to $S_1 - S_2$.

2. *Complementation*: Let $S \subseteq \mathbf{R}^n$ have Property SL. Obviously, $\mathbf{R}^n$ satisfies Property SL. Hence, the complement of S, $\mathbf{R}^n - S$, satisfies Property SL.

3. *Intersection*: Let $S_1, S_2 \subseteq \mathbf{R}^n$ both have Property SL. Then $S_1 \cap S_2 = S_1 - (\mathbf{R}^n - S_2)$ has Property SL.

4. *Union*: Let $S_1, S_2 \subseteq \mathbf{R}^n$ both have Property SL. Then $S_1 \cup S_2 = \mathbf{R}^n - ((\mathbf{R}^n - S_1) \cap (\mathbf{R}^n - S_2))$ has Property SL. ∎

Lemma A.2 *Having* Property SL *is preserved under taking topological boundary, closure, and interior.*

Proof.

1. *Closure*: Let $S \subseteq \mathbf{R}^n$ have Property SL. To show that $\overline{S}$ has Property SL, let $\vec{p}$ be in $\overline{\overline{S}} = \overline{S}$. Let V be a convex neighborhood of $\vec{p}$ showing that S has Property SL at $\vec{p}$. Let $\vec{q}$ be in V. We consider both possibilities:

   (a) $\vec{q}$ *is in* $\overline{S}$: Then there must exist a sequence $(\vec{q_m})_{m\geq 0}$ in $S \cap V$ such that $\lim_{m\to\infty} \vec{q}_m = \vec{q}$. Since S has Property SL, we have, for all $m \geq 0$, that $]\vec{p}, \vec{q_m}[$ is fully contained within S, whence, in the limit, $]\vec{p}, \vec{q}[$ is fully contained within $\overline{S}$.

   (b) $\vec{q}$ *is not in* $\overline{S}$: We have to show that $]\vec{p}, \vec{q}[$ is disjoint from $\overline{S}$. Therefore, assume to the contrary that both sets have the point $\vec{r}$ in common. Since $\vec{r}$ is in $\overline{S}$, there must exist a sequence $(\vec{r}_m)_{m\geq 0}$ in S such that $\lim_{m\to\infty} \vec{r}_m = \vec{r}$. Since $\vec{r}$ is on the open line segment between $\vec{p}$ and $\vec{q}$, there exists $\lambda > 0$ such that $\vec{q} = \vec{p} + \lambda(\vec{r} - \vec{p})$. Let, for $m \geq 0$, $\vec{q}_m = \vec{p} + \lambda(\vec{r}_m - \vec{p})$. Obviously, $\lim_{m\to\infty} \vec{q}_m = \vec{q}$. Since $\vec{q}$ is not in $\overline{S}$, there must exist $m_0 \geq 0$ such that $\vec{q}_{m_0}$ is not in S. Since S has Property SL, we have that $]\vec{p}, \vec{q}_{m_0}[$ is disjoint from S. Hence, $\vec{r}_{m_0}$, which by construction is on this line segment, does not belong to S, a contradiction.

2. *Boundary*: Let $S \subseteq \mathbf{R}^n$ have Property SL. The topological boundary of S equals $\overline{S} \cap \overline{\mathbf{R}^n - S}$, and, therefore, has Property SL.

3. *Interior*: Let $S \subseteq \mathbf{R}^n$ have Property SL. The topological interior of S equals $\mathbf{R}^n - \overline{\mathbf{R}^n - S}$, and, therefore, has Property SL. ∎

Using some of the above preservation properties, we can easily show the "only if" of Proposition 4.2:

Proposition A.3 *Each* **R***-semi-linear set has* Property SL*.*

Proof. First, we observe that closed half-spaces obviously have Property SL. As a consequence, convex polyhedra must have Property SL, by Lemma A.1, and open convex polyhedra must have Property SL, by Lemma A.2. Finally, each **R**-semi-linear set, which is a finite union of open convex polyhedra (see, e.g., [73]), must have Property SL, again by Lemma A.1. ∎

We now come to the "if" part of Proposition 4.2, which is the more interesting direction of this result.

First, we prove Proposition 4.5:

Proposition 4.5 *A* bounded *set having* Property SL *is* **R***-semi-linear.*

Proof. Let $S \subseteq \mathbf{R}^n$ be a bounded set. We show by induction on the dimension of S that S is **R**-semi-linear. If S is zero-dimensional, then S consists of a finite number of points, whence S is trivially **R**-semi-linear. Now, assume that, for some k, $0 \leq k < n$, each bounded subset of $\mathbf{R}^n$ with Property SL and dimension at most k is **R**-semi-linear, and let S be $(k+1)$-dimensional. For $\vec{p} \in \mathbf{R}^n$ and $A \subseteq \mathbf{R}^n$, let us define the cone from $\vec{p}$ on A as the union of all closed line segments connecting $\vec{p}$ to a point of A. By Property SL, there exists, for each point $\vec{p}$ of $\overline{S}$, a neighborhood $V_{\vec{p}}$, which we can assume to be an n-dimensional hypercube, such that $S \cap V_{\vec{p}}$ is either the cone from $\vec{p}$ on $S \cap \partial V_{\vec{p}}$, if $\vec{p}$ is in S, or that set with $\vec{p}$ removed, if $\vec{p}$ is not in S. Since $\partial V_{\vec{p}}$, the boundary of the hypercube $V_{\vec{p}}$, is obviously **R**-semi-linear, it has Property SL (see also Lemma A.3 in Appendix A). Since, clearly, the intersection of two sets with Property SL has Property SL (see also Lemma A.1 in Appendix A), $S \cap \partial V_{\vec{p}}$ has Property SL. Since, moreover, this set is at most k-dimensional, the induction hypothesis applies, whence it is **R**-semi-linear. Since $S \cap \partial V_{\vec{p}}$ is **R**-semi-linear, it follows from Proposition 2.12 that it is the finite union of convex polyhedra. Clearly, the cone from a point on a convex polyhedron equals the convex closure of that point and that polyhedron, whence that cone is **R**-semi-linear. Hence, since the cone defining $S \cap V_{\vec{p}}$ equals the finite union of the cones from $\vec{p}$ on a polyhedron defining $S \cap \partial V_{\vec{p}}$, possibly with the point $\vec{p}$ removed, $S \cap V_{\vec{p}}$ is **R**-semi-linear. Clearly, $\overline{S} \subseteq \bigcup_{\vec{p} \in \overline{S}} V_{\vec{p}}$. Since $\overline{S}$ is both closed and bounded, it is

compact[1], whence there exist finitely many points of $\overline{S}$, say $\vec{p}_1, \ldots, \vec{p}_m$, such that $\overline{S} \subseteq \bigcup_{i=1}^{m} V_{\vec{p}_i}$. Consequently, $S = \bigcup_{i=1}^{m} (S \cap V_{\vec{p}_i})$ is a finite union of **R**-semi-linear sets, and, therefore, **R**-semi-linear. ∎

The above proof only works for the bounded case, as it uses compactness, and is highly non-constructive, for the same reason.

We now prove Property SL for the general case, but, first, we state and prove the following two lemmas.

Lemma A.4 *A regular stratum (of the first layer of the regular stratification) of an* **R***-semi-algebraic set having* Property SL *is contained in an affine subspace within which it is topologically open.*

Proof. Let $S \subseteq \mathbf{R}^n$ be an **R**-semi-algebraic set having Property SL, and let S_i be a regular stratum of the first layer of the regular stratification of S.

Let $\vec{p}$ be a point in S_i. By Definition 3.12, we know there exist polynomials with real coefficients $P_1, \ldots, P_k$ in real variables, $k = n - dim(S)$, such that

$$\frac{\mathrm{d}P_1}{\mathrm{d}\vec{x}}(\vec{p}), \ \ldots, \ \frac{\mathrm{d}P_k}{\mathrm{d}\vec{x}}(\vec{p})$$

are linearly independent and

$$S \cap V = \{\vec{x} \in V \mid P_1(\vec{x}) = \cdots = P_k(\vec{x}) = 0\}. \tag{A.1}$$

Let T be the $(n-k)$-dimensional tangent space of S at $\vec{p}$, defined by the system of k linear equations

$$\frac{\mathrm{d}P_1}{\mathrm{d}\vec{x}}(\vec{p}) \,.\, \vec{x} = 0, \ \ldots, \ \frac{\mathrm{d}P_k}{\mathrm{d}\vec{x}}(\vec{p}) \,.\, \vec{x} = 0. \tag{A.2}$$

Without loss of generality, we may assume that V shows that S has Property SL at $\vec{p}$.

By the Implicit Function Theorem, there exist k coordinate positions, say the first k for simplicity of notation, there exists an open convex neighborhood W_1 of $(p_1, \ldots, p_k)$ in $\mathbf{R}^k$ and an open convex neighborhood W_2 of $(p_{k+1}, \ldots, p_n)$ in $\mathbf{R}^{n-k}$ with $W = W_1 \times W_2 \subseteq V$, and there exist analytic functions $f_1, \ldots, f_k$ such that

$$\begin{aligned} S \cap W = \{\vec{x} \mid\ & x_1 = f_1(x_{k+1}, \ldots, x_n), \ \ldots, \ x_k = f_k(x_{k+1}, \ldots, x_n), \\ & (x_{k+1}, \ldots, x_n) \in W_2\}. \end{aligned} \tag{A.3}$$

[1] A subset of a topological space is called *compact* if each cover by open sets can be reduced to a finite cover. A subset of $\mathbf{R}^n$ is compact precisely if it is closed and bounded.

Consequently, the system of k linear equations A.2 defining T can be rewritten as

$$\left\{\begin{array}{rcl} x_1 & = & \frac{\partial f_1}{\partial x_{k+1}}(p_{k+1},\ldots,p_n)\,.\,x_{k+1}+\cdots+\frac{\partial f_1}{\partial x_n}(p_{k+1},\ldots,p_n)\,.\,x_n, \\ \vdots & & \vdots \\ x_k & = & \frac{\partial f_k}{\partial x_{k+1}}(p_{k+1},\ldots,p_n)\,.\,x_{k+1}+\cdots+\frac{\partial f_k}{\partial x_n}(p_{k+1},\ldots,p_n)\,.\,x_n. \end{array}\right. \tag{A.4}$$

Now, let $\vec{q}$ be any point of $S \cap W$ different from $\vec{p}$. By Property SL, the convexity of W, and the defining properties of $\vec{p}$ and $\vec{q}$, $[\vec{p},\vec{q}]$ is fully contained within $S \cap W$. Necessarily, a line segment through $\vec{p}$ which is fully contained in S is also fully contained in the tangent space T of S at $\vec{p}$, whence in particular $\vec{q}$ is in T. Thus, we have shown that $S \cap W$ is fully contained in T. We next show that, also, $T \cap W$ is fully contained in S. Thereto, let $\vec{r}$ be any point of $T \cap W$. Then $(r_{k+1},\ldots,r_n)$ is in W_1. By (3), there is a unique point $\vec{s}$ in $S \cap W$ for which $s_{k+1} = r_{k+1},\ldots,s_n = r_n$. Since $S \cap W$ is fully contained in T, it follows that $\vec{s}$ is in T. By (4), $\vec{r}$ is the only point of T for which $r_{k+1} = s_{k+1},\ldots,r_n = s_n$. Hence $\vec{r} = \vec{s}$, whence $\vec{r}$ is in S. We may thus conclude that $S \cap W = T \cap W$, as a consequence of which all points in $S \cap W$ are regular and must belong to S_i. Another consequence of $S \cap W = T \cap W$ is that $S \cap W$ is topologically open within T.

By analytic continuation, i.e., by making the above argument for all points $\vec{p}$ in S_i and by considering that the constructed neighborhoods overlap, we see that (i) S_i is fully contained in T, and that (ii) S_i is topologically open within T. ∎

Lemma A.5 *A regular stratum (of the first layer of the regular stratification) of an* **R***-semi-algebraic set having* Property SL *is an* **R***-semi-algebraic set which also has* Property SL*.*

Proof. Let $S \subseteq \mathbf{R}^n$ be an **R**-semi-algebraic set having Property SL, and let S_i be a regular stratum of the first layer of the regular stratification of S. It is well-known that S_i is **R**-semi-algebraic [46, 56].

By Lemma A.4, there exists an affine subspace T of $\mathbf{R}^n$ such that S_i is contained in T and S_i is topologically open within T. Obviously, $\overline{S_i}$ is also contained in T. We show that S_i has Property SL.

Thereto, let $\vec{p}$ be a point of $\overline{S_i}$.

If $\vec{p}$ is a point of S_i, then there exists a neighborhood V of $\vec{p}$ such that $S \cap V = S_i \cap V = T \cap V$. Using this neighborhood V, it is immediately seen that S_i has Property SL at $\vec{p}$. Thus suppose $\vec{p}$ is not in S_i. Let V be a convex neighborhood of $\vec{p}$ showing that S has Property SL at $\vec{p}$. Let $\vec{q}$ be in V.

We distinguish two cases:

1. *The point $\vec{q}$ is in S_i*: Then there exists a neighborhood $W \subseteq V$ of $\vec{q}$ such that $S \cap W = S_i \cap W = T \cap W$. Let $\vec{r}$ be a point on $]\vec{p}, \vec{q}[$. Then there exists λ, $0 < \lambda < 1$, such that $\vec{r} = \vec{p} + \lambda(\vec{q} - \vec{p})$. Let $X = \{\vec{p} + \lambda(\vec{s} - \vec{p}) \mid \vec{s} \in W\}$. Then X is a neighborhood of $\vec{r}$. By applying Property SL, it is easily seen that $S \cap X = T \cap X$, whence $\vec{r}$ is regular and S has maximal dimension at $\vec{r}$. Thus, $S_i \cup]\vec{p}, \vec{q}[$, which is connected, is a superset of S_i consisting only of regular points of S in which S has maximal dimension. By definition of regular stratum, $S_i \cup]\vec{p}, \vec{q}[= S_i$, whence $]\vec{p}, \vec{q}[$ is fully contained within S_i.

2. *The point $\vec{q}$ is not in S_i*: If $\vec{q}$ is not in S, then, by Property SL, $]\vec{p}, \vec{q}[$ is disjoint from S, whence disjoint from S_i. Similarly, if $\vec{q}$ is not in T, $]\vec{p}, \vec{q}[$ is disjoint from T, whence disjoint from S_i. Thus suppose $\vec{q}$ is in $S \cap T$.

   We again distinguish two cases:

   (a) *Each neighborhood of $\vec{q}$ contains points of S outside T.* Then there must exist a sequence of points $(\vec{q}_m)_{m \geq 0}$ in $S \cap V$ not belonging to T such that $\lim_{m \to \infty} \vec{q}_m = \vec{q}$. Let $\vec{r}$ be on $]\vec{p}, \vec{q}[$. Then there exists λ, $0 < \lambda < 1$, such that $\vec{r} = \vec{p} + \lambda(\vec{q} - \vec{p})$. Let, for $m \geq 0$, $\vec{r}_m = \vec{p} + \lambda(\vec{q}_m - \vec{p})$. Clearly, $\vec{r}_m$ is not in T, and $\lim_{m \to \infty} \vec{r}_m = \vec{r}$, whence each neighborhood of $\vec{r}$ contains points of S outside T. Hence, $\vec{r}$ cannot belong to S_i. We have thus proved that $]\vec{p}, \vec{q}[$ is disjoint from S_i.

   (b) *Some neighborhood of $\vec{q}$ contains no points of S outside T.* Let $W \subseteq V$ be such a neighborhood; thus, $S \cap W = T \cap W$, and $\vec{q}$ is a regular point of S in which S has maximal dimension. Let $S_j \neq S_i$ be the regular stratum to which $\vec{q}$ belongs. By an argument similar to the one used in Case 1, we can show that the open line segment between $\vec{p}$ and $\vec{q}$ is fully contained in S_j, whence it is disjoint from S_i.

We may thus conclude that S_i has Property SL. ∎

Lemma A.6 *A set which is topologically open within its affine support, and whose topological boundary within its affine support can be decomposed into a disjoint union of sets each of which is topologically open within its affine support, is* **R***-semi-linear.*

Proof. Let $S_0 \subseteq \mathbf{R}^n$. Let T_0 be the affine support of S_0 and assume that S_0 is open within T_0. Let ∂S_0 be the topological boundary of S_0 within T_0. Assume furthermore that ∂S_0 is the disjoint union of $S_1, \ldots, S_m$, that $T_1, \ldots, T_m$ are the affine supports of $S_1, \ldots, S_m$, respectively, and that, for $i = 1, \ldots, m$, S_i is topologically open within T_i. We prove that S_0 is **R**-semi-linear.

For $i = 0, \ldots, m$, let $H_{i1}, \ldots, H_{ik_i}$ be $(n-1)$-dimensional hyperplanes the intersection of which equals T_i. For $i = 0, \ldots, m$ and $j = 1, \ldots k_i$, let $\mathcal{P}_{ij}$ be the partition of $\mathbf{R}^n$ consisting of the hyperplane H_{ij} and the two open half-spaces it delineates. Let $\mathcal{P}$ be the coarsest common refinement of $\mathcal{P}_{ij}$, $0 \leq i \leq m$ and $1 \leq j \leq k_i$. Clearly, each cell[2] of $\mathcal{P}$ can be obtained by choosing, for each $i = 0, \ldots, m$ and for each $j = 1, \ldots, k_i$, one cell of $\mathcal{P}_{ij}$, and then taking the intersection of the chosen cells. Hence, $\mathcal{P}$ is a finite partition of $\mathbf{R}^n$ whose cells are open convex polyhedra and, therefore, $\mathbf{R}$-semi-linear.

Thus, it suffices to show that S_0 is a union of such cells.

Thereto, let $\vec{p}$ be a point of S_0. For $i = 0, \ldots, m$ and $j = 1, \ldots, k_i$, let $H_{ij}^{\vec{p}}$ be either H_{ij}, if $\vec{p}$ is in H_{ij}, or the open half-space delineated by H_{ij} that contains $\vec{p}$. Then,

$$\vec{p} \in \bigcap_{i=0}^{m} \bigcap_{j=1}^{k_i} H_{ij}^{\vec{p}}.$$

By the observation above, the right-hand side of the above containment is the cell of $\mathcal{P}$ containing $\vec{p}$, which we shall denote as $N_{\vec{p}}$. Notice that $N_{\vec{p}} \subseteq T_0$. Now suppose that $N_{\vec{p}}$ also contains a point, say $\vec{q}$, not belonging to S_0. By the convexity of $N_{\vec{p}}$, $[\vec{p}, \vec{q}]$ is fully contained within $N_{\vec{p}}$. Since S_0 is open in T_0, there exists a point $\vec{r} \neq \vec{p}$ on $[\vec{p}, \vec{q}]$ such that (i) $[\vec{p}, \vec{r}[$ is fully contained within S_0, and (ii) $\vec{r}$ is not in S_0. Consequently, $\vec{r}$ is in ∂S_0. Let S_i, $1 < i \leq m$, be the set in the decomposition of ∂S_0 to which $\vec{r}$ belongs. Clearly, S_i and S_0 are disjoint. Moreover, $\vec{p}$ is not in T_i. Indeed, if $\vec{p}$ were in T_i, then $[\vec{p}, \vec{r}[$ would be fully contained in T_i. Since S_i is topologically open within T_i, it follows that $[\vec{p}, \vec{r}[$ would intersect S_i. Hence, $[\vec{p}, \vec{r}[$ would contain points outside S_0, a contradiction. Since $\vec{p}$ is not in T_i, there exists j, $1 \leq j \leq k_i$ such that $\vec{p}$ is not in H_{ij}. Of course, $\vec{r}$ is in H_{ij} and, therefore, not in $H_{ij}^{\vec{p}}$, whence not in $N_{\vec{p}}$, a contradiction. We may thus conclude that $N_{\vec{p}}$ is fully contained in S_0.

Finally, $S_0 = \bigcup_{\vec{p} \in S_0} N_{\vec{p}}$. ∎

We are now ready to prove the "if" part of Proposition 4.2.

Proposition A.7 *Each* $\mathbf{R}$*-semi-algebraic set with* Property SL *is* $\mathbf{R}$*-semi-linear.*

Proof. Let $S \subseteq \mathbf{R}^n$ be an $\mathbf{R}$-semi-algebraic set with Property SL.

Let $S_1, \ldots, S_l$ be the (pairwise disjoint) regular strata of the first layer of the regular stratification of S. By Lemmas A.4 and A.5, there exist affine subspaces $T_1, \ldots, T_l$

[2]We use the term *cell* to refer to an element of a partition (which is a subset of the set being partitioned).

of $\mathbf{R}^n$ such that, for $i = 1, \ldots, l$, (i) S_i is an $\mathbf{R}$-semi-algebraic subset of T_i which is topologically open within T_i, and (ii) S_i has Property SL. By Lemma A.1, $S - \bigcup_{i=1}^{l} S_i$ is an $\mathbf{R}$-semi-algebraic set having Property SL. We can thus recursively do the same procedure on $S - \bigcup_{i=1}^{l} S_i$ to obtain the regular strata of the subsequent layers of the regular stratification of S. Since $dim(S - \bigcup_{i=1}^{l} S_i) < dim(S)$, the recursion stops, and we obtain a decomposition of S as a disjoint union of sets $S_1, \ldots, S_m$, for which there exist affine subspaces $T_1, \ldots, T_m$ of $\mathbf{R}^n$ such that, for $i = 1, \ldots, m$, (i) S_i is an $\mathbf{R}$-semi-algebraic subset of T_i which is topologically open within T_i, and (ii) S_i has Property SL. It suffices to show that $S_1, \ldots, S_m$ are $\mathbf{R}$-semi-linear.

Thereto, let $1 \leq i \leq m$, and consider S_i. By Lemma A.2, ∂S_i, the topological boundary of S_i within T_i, which is also $\mathbf{R}$-semi-algebraic, has Property SL. Thus, we can repeat the above reasoning and decompose ∂S_i as a disjoint union of sets each of which is topologically open within its affine support (and which has Property SL). It now follows immediately from Lemma A.6 that S_i is $\mathbf{R}$-semi-linear. ■

From Propositions A.3 and A.7, our final result is now immediate:

Proposition 4.2 Let S be an $\mathbf{R}$-semi-algebraic set. The set S is $\mathbf{R}$-semi-linear if and only if it has Property SL.

Appendix B

Alternative Proof for Decidability of Semi-Linearity for Semi-Algebraic Sets

In this appendix, we present an alternative method, which was kindly suggested by an anonymous referee of the extended abstract of the paper [20], presented at the 16th *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, to prove the two principal decidability results in Chapter 4 (Theorems 4.37 and 4.40).

This alternative method is based on the well-known fact that the real ordered field $\langle \mathbf{R}, \leq, \times, +, 0, 1 \rangle$ and the real ordered group $\langle \mathbf{R}, \leq, +, 0, 1 \rangle$ have "definable choice functions": for any definable set, one can definably choose an element in it (e.g., [11]). More formally, if $\varphi(\vec{x})$ is a satisfiable real formula, i.e., $S = \{\vec{x} \mid \varphi(\vec{x})\}$ is not empty, one can construct a real formula $\varphi'(\vec{x})$ such that $S' = \{\vec{x} \mid \varphi'(\vec{x})\}$ is a singleton subset of S. Moreover, it is possible to construct this formula in such a way that (i) $\varphi'(\vec{x})$ is $\mathbf{Z}$-linear if $\varphi(\vec{x})$ is $\mathbf{Z}$-linear, and (ii), whenever $\varphi(\vec{x})$ and $\psi(\vec{x})$ define the same set S, $\varphi'(\vec{x})$ and $\psi'(\vec{x})$ define the same singleton subset S' of S. In geometric terms, the above translates as follows: in a non-empty semi-algebraic set, one can choose deterministically a point with real algebraic coordinates in an algorithmic manner; if, moreover, the set is $\mathbf{Z}$-semi-linear, then the point has rational coordinates.[1]

[1]To see this, let S be a semi-algebraic set. If S is discrete (which is decidable by the same

Using this result, we prove Theorems 4.37 and 4.40.

Theorem 4.37 *Let S be a semi-algebraic set. The set S is* **A***-semi-linear if and only if it has* Property SL. *Moreover,* **A***-semi-linearity of a semi-algebraic set defined by a real formula is decidable.*

Proof. By Proposition 4.2, it suffices to show that, if S is a semi-algebraic **R**-semi-linear set, then S is **A**-semi-linear. Since S is semi-algebraic, there exists a real formula $\varphi(\vec{x})$ defining S. Since S is **R**-semi-linear, there also exists an **R**-linear formula $\psi(\vec{x})$ defining S. Let $\theta(\vec{\lambda}, \vec{x})$ be the real formula obtained from ψ by replacing each coefficient in ψ by a real variable. Hence, there exists a vector $\vec{r}$ of real numbers such that $\psi(\vec{x}) \equiv \theta(\vec{r}, \vec{x})$. Next, let $\gamma(\vec{\lambda})$ be the formula $(\forall \vec{x})(\varphi(\vec{x}) \Leftrightarrow \theta(\vec{\lambda}, \vec{x}))$. This formula is satisfiable, since $\gamma(\vec{r})$ is *true*. Let $\vec{a}$ be the unique vector of real algebraic numbers such that $\gamma'(\vec{a})$ is *true*. Clearly, the formula $\tilde{\psi}(\vec{x}) \equiv \theta(\vec{a}, \vec{x})$ is an **A**-semi-linear formula defining S. ∎

Theorem 4.40 *It is decidable whether a semi-algebraic set defined by a real formula is* **Z***-semi-linear.*

Proof. Assuming that we have already decided that S is **A**-semi-linear, we produce by enumeration an **A**-linear formula $\varphi(\vec{x})$ defining S. Let $\theta(\vec{\lambda}, \vec{x})$ be the real formula obtained from φ by replacing each coefficient in φ by a real variable, and let $\vec{a}$ be the vector of real algebraic numbers such that $\varphi(\vec{x}) \equiv \theta(\vec{a}, \vec{x})$.

Let $\gamma(\vec{\lambda})$ be the formula $(\forall \vec{x})(\varphi(\vec{x}) \Leftrightarrow \theta(\vec{\lambda}, \vec{x}))$. This formula is satisfiable, since $\gamma(\vec{a})$ is *true*. Let $\vec{q}$ be the unique vector of real algebraic numbers such that $\gamma'(\vec{q})$ is *true*. Notice that the formula $\gamma'(\vec{\lambda})$ can effectively be computed.

We now claim that S is **Z**-semi-linear if and only if $\vec{q}$ is a vector of rational numbers. Lemma 4.39 then yields the desired result.

Clearly, if $\vec{q}$ is a vector of rational numbers, then $\tilde{\varphi}(\vec{x}) \equiv \theta(\vec{q}, \vec{x})$ is a **Z**-linear formula defining S, whence S is **Z**-semi-linear.

Conversely, if S is **Z**-semi-linear, then let $\psi(\vec{x})$ be a **Z**-semi-linear formula defining S. Consider the formula $(\forall \vec{x})(\psi(\vec{x}) \Leftrightarrow \theta(\vec{\lambda}, \vec{x}))$. Using quantifier elimination [19], it

expression as for semi-linear sets [74]), then choose a point of S based on minimality of coordinates. From the observation that **Z**-linear formula permit quantifier elimination [28, 54, 48], it is readily seen that, if S is **Z**-semi-linear, this point will have rational coordinates. If S is not discrete, then at least one of its projections on the coordinate axes is not discrete. Let S_i be the projection on the first such coordinate axis (according to the standard order of the coordinates), say x_i. Since S_i is not discrete, S_i contains an open interval, and, therefore, a point whose coordinate is a rational number. Let q be the first such rational number in some enumeration of the rational numbers. The intersection of S and the hyperplane $x_i = q$ has strictly lower dimension than S and is **Z**-semi-linear whenever S is. The procedure above is now repeated for this lower dimensional set.

can be seen that the above formula is equivalent to a **Z**-linear formula $\chi(\vec{\lambda})$. Hence, $\chi'(\vec{\lambda})$ is also **Z**-linear. Since $\gamma(\vec{\lambda})$ and $\chi(\vec{\lambda})$ clearly define the same semi-algebraic set, it follows that $\gamma'(\vec{\lambda})$ and $\chi'(\vec{\lambda})$ both define $\vec{q}$. Since $\chi'(\vec{q})$ is *true*, it follows that $\vec{q}$ is a vector of rational numbers. ■

Bibliography

[1] D. Abel and B.C. Ooi, editors. *Proceedings of the 3rd International Symposium on Spatial Database*, volume 692 of *Lecture Notes in Computer Science*, Berlin, 1993. Springer-Verlag.

[2] F. Afrati, T. Andronikos, and T. Kavalieros. On the expressiveness of first-order constraint languages. In G. Kuper and M. Wallace, editors, *Proceedings of the 1st Workshop on Constraint Databases and Applications*, volume 1034 of *Lecture Notes in Computer Science*, pages 22–39, Berlin, 1995. Springer-Verlag.

[3] F. Afrati, T. Andronikos, and T. Kavalieros. On the expressiveness of query languages with linear constraints: Capturing desirable spatial properties. In V. Gaede, A. Brodsky, O. Günther, 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 105–115, Berlin, 1997. Springer-Verlag.

[4] F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper. Linear versus polynomial constraints in database query languages. In A. Borning, editor, *Proceedings of the 2nd International Workshop on Principles and Practice of Constraint Programming*, volume 874 of *Lecture Notes in Computer Science*, pages 181–192, Berlin, 1994. Springer-Verlag.

[5] W.G. Aref and H. Samet. Extending a database with spatial operations. In O. Günther and H.-J. Schek, editors, *Proceedings of the 2nd International Symposium on Spatial Database*, volume 525 of *Lecture Notes in Computer Science*, pages 299–319, Berlin, 1991. Springer-Verlag.

[6] D.S. Arnon. Geometric reasoning with logic and algebra. *Artificial Intelligence*, 37:37–60, 1988.

[7] C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa. The QuickHull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[8] M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational Expressive Power of Constraint Query Languages. *Journal of the ACM*, 45(1):1–34, 1998.

[9] M. Benedikt and L. Libkin. Languages for relational databases over interpreted structures. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 87–98, New York, 1997. ACM Press.

[10] M. Benedikt and L. Libkin. Safe constraint queries. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 99–108, New York, 1998. ACM Press.

[11] J. Bochnak, M. Coste, and M.F. Roy. *Real Algebraic Geometry.* Springer-Verlag, Berlin, 1987.

[12] A. Brodsky, V.E. Segal, J. Chen, and P.A. Exarkhopoulo. The CCUBE constraint object-oriented database system. *Constraints Journal*, 2(3/4):245–277, 1997.

[13] A. Buchmann, editor. *Proceedings of the 1st International Symposium on Spatial Database*, volume 409 of *Lecture Notes in Computer Science*, Berlin, 1989. Springer-Verlag.

[14] J.-H. Byon and P. Revesz. Disco: a constraint database system with sets. In G. Kuper and M. Wallace, editors, *Proceedings Workshop on Constraint Databases and Applications (Friedrichshafen, Germany)*, volume 1034 of *Lecture Notes in Computer Science*, pages 68–83, Berlin, 1996. Springer-Verlag.

[15] A. Chandra and D. Harel. Computable queries for relational database systems. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.

[16] J. Chomicki, D.Q. Goldin, and G.M. Kuper. Variable independence and aggregation closure. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 40–48, New York, 1996. ACM Press.

[17] E. Clementini, P. Di Felice, and P. van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In D. Abel and B.C. Ooi, editors, *Proceedings of the 3rd International Symposium on Spatial Database*, volume 692 of *Lecture Notes in Computer Science*, pages 277–295, Berlin, 1993. Springer-Verlag.

[18] E.F. Codd. A relational model of data for large shared data banks (reprint). *Communications of the ACM*, 26(1):64–69, 1983.

[19] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, Berlin, 1975. Springer-Verlag.

[20] F. Dumortier, M. Gyssens, L. Vandeurzen, and D. Van Gucht. On the decidability of semi-linearity for semi-algebraic sets and its implications for spatial databases. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 68–77, New York, 1997. ACM Press. Full version to appear in *Journal of Computer and System Sciences.*

[21] M.J. Egenhofer. A formal definition of binary topological relationships. In W. Litwin and H.-J. Schek, editors, *Proceedings Foundations of Data Organization and Algorithms*, volume 367 of *Lecture Notes in Computer Science*, pages 457–472, Berlin, 1989. Springer-Verlag.

[22] M.J. Egenhofer. Reasoning about binary topological relations. In O. Günther and H.-J. Schek, editors, *Proceedings of the 2nd International Symposium on Spatial Database*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160, Berlin, 1991. Springer-Verlag.

[23] M.J. Egenhofer. Why not SQL! *International Journal on Geographical Information Systems*, 6(2):71–85, 1992.

[24] M.J. Egenhofer. What is special about spatial? Database requirements for vehicle navigation in geographic space. In *Proceedings of the 13th ACM SIGMOD International Conference on Management of Data*, pages 398–402, New York, 1993. ACM Press.

[25] M.J. Egenhofer. Spatial SQL: a query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[26] M.J. Egenhofer and J. Herring. A mathematical framework for the definition of topological relationships. In K. Brassel and H. Kishimoto, editors, *Proceedings 4th International Symposium on Spatial Data Handling (Zurich, Switzerland)*, pages 803–813, 1990.

[27] M.J. Egenhofer and J.R. Herring, editors. *Proceedings of the 4th International Symposium on Spatial Database*, volume 951 of *Lecture Notes in Computer Science*, Berlin, 1995. Springer-Verlag.

[28] J. Ferrante and C. Rackoff. A decision procedure for the first-order theory of real addition with order. *SIAM Journal of Computing*, 4(1):69–76, 1975.

[29] K. Fukuda and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, *Combinatorics and computer science*, volume 1120 of *Lecture Notes in Computer Science*, pages 91–111, Berlin, 1996. Springer-Verlag.

[30] G. Goos and J. Hartmanis, editors. *Efficient Query Processing in Geographic Information Systems*, volume 471 of *Lecture Notes in Computer Science*, Berlin, 1990. Springer-Verlag.

[31] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. DEDALE, a spatial constraint database. In *Proceedings of the 6th International Workshop on Database Programming Languages*, volume 1369 of *Lecture Notes in Computer Science*, pages 38–59, Berlin, 1997. Springer-Verlag.

[32] S. Grumbach, P. Rigaux, and L. Segouffin. The DEDALE system for complex spatial queries. In L. Haas and A. Tiwary, editors, *Proceedings of the 17th ACM SIGMOD International Conference on Management of Data*, pages 213–224, New York, 1998. ACM Press.

[33] S. Grumbach and J. Su. Finitely representable databases. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 289–300, New York, 1994. ACM Press.

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

[35] S. Grumbach, J. Su, and C. Tollu. Linear constraint query languages: Expressive power and complexity. In D. Leivant, editor, *Proceedings of the Logic and Computational Complexity Workshop '94*, volume 960 of *Lecture Notes in Computer Science*, pages 426–446, Berlin, 1995. Springer-Verlag.

[36] O. Günther, editor. *Efficient Structures for Geometric Data Management*, volume 337 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1988.

[37] O. Günther. Research issues in spatial databases. *SIGMOD Record*, 19(4):61–68, 1990.

[38] O. Günther and H.-J. Schek, editors. *Proceedings of the 2nd International Symposium on Spatial Database*, volume 525 of *Lecture Notes in Computer Science*, Berlin, 1991. Springer-Verlag.

[39] R.H. Güting. Geo-relational algebra: a model and query language for geometric database systems. In J.W. Schmidt, S. Ceri, and M. Missikoff, editors, *Advances in Database Technology—EDBT '88, Proceedings International Conference on*

Extending Database Technology, volume 303 of *Lecture Notes in Computer Science*, pages 506–527, Berlin, 1988. Springer-Verlag.

[40] R.H. Güting. Gral: An extensible relational database system for geometric applications. In *Proceedings of the 15th International Conference on Very Large Databases*, pages 33–44, 1989.

[41] R.H. Güting. An introduction to spatial database systems. *VLDB Journal*, 3(4):357–399, 1994.

[42] R.H. Güting, T. de Ridder, and M. Schneider. Implementation of the rose algebra: efficient algorithms for realm-based spatial data types. In M.J. Egenhofer and J.R. Herring, editors, *Proceedings of the 4th International Symposium on Spatial Database*, volume 951 of *Lecture Notes in Computer Science*, pages 216–239, Berlin, 1995. Springer-Verlag.

[43] R.H. Güting and M. Schneider. Realms: a foundation for spatial data types in database systems. In *Proceedings 3rd International Conference on Very Large Databases (Singapore)*, pages 14–35, 1993.

[44] R.H. Güting and M. Schneider. Realm-based spatial data types: The rose algebra. *VLDB Journal*, 4(2):243–286, 1995.

[45] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometrical query languages. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 62–67, New York, 1997. ACM Press. Full version to appear in *Journal of Computer and System Sciences.*

[46] J. Heintz, T. Recio, and M.F. Roy. Algorithms in real algebraic geometry and applications to computational geometry. In *Discrete and Computational Geometry*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 137–163. AMS-ACM, 1991.

[47] J.E. Hopcroft and J.D Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[48] T. Huynh, C. Lassez, and J.-L. Lassez. Fourier algorithm revisited. In H. Kirchner and W. Wechler, editors, *Proceedings 2nd Int'l Conf. on Algebraic an Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 117–131, Berlin, 1990. Springer Verlag.

[49] P.C. Kanellakis and D.Q. Goldin. Constraint programming and database query languages. In M. Hagiya and J.C. Mitchell, editors, *Proceedings 2nd Conference on Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 96–120, Berlin, 1994. Springer-Verlag.

[50] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[51] A. Kemper and M. Wallrath. An analysis of geometric modeling in database systems. *Computing Surveys*, 19(1):47–91, 1987.

[52] B. Kuijpers, J. Paredaens, and L. Vandeurzen. Semantics in spatial databases. In B. Thalheim, editor, *Proceedings of the Workshop Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 114–134, Berlin, 1995. Springer-Verlag.

[53] G. Kuper. On the expressive power of the relational calculus with arithmetic constraints. In S. Abiteboul and P.C. Kanellakis, editors, *Proceedings of the 3rd International Conference on Database Theory*, volume 470 of *Lecture Notes in Computer Science*, pages 202–214, Berlin, 1990. Springer-Verlag.

[54] J. L. Lassez. Querying constraints. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 288–298, New York, 1990. ACM Press.

[55] C. B. Medeiros and F. Pires. Databases for GIS. *SIGMOD Record*, 23(1):107–115, 1994.

[56] T. Mostowski and E. Rannou. Complexity of the computation of the canonical Whitney stratification of an algebraic set in $\mathbf{C}^n$. In H.F. Mattson, T. Mora, and R.R.N. Rao, editors, *Proceedings 9th International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, volume 539 of *Lecture Notes in Computer Science*, pages 281–291, Berlin, 1991. Springer-Verlag.

[57] T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games II*, volume 8 of *Annals of Mathematical Studies*, pages 51–73, Princeton, 1953. Princeton University Press.

[58] J. Nievergelt and M. Freeston. Special issue on spatial data. *Computer Journal*, 37(1), 1994.

[59] F. Van Oystaeyen. Personal communication, 1999.

[60] J. Paredaens. Spatial databases, the final frontier. In G. Gottlob and M.Y. Vardi, editors, *Proceedings of the 5th International Conference on Database Theory*, volume 893 of *Lecture Notes in Computer Science*, pages 14–32, Berlin, 1995. Springer-Verlag.

[61] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 279–288, New York, 1994. ACM Press.

[62] J. Paredaens and B. Kuijpers. Data models and query languages for spatial databases. *Data and Knowledge Engineering*, 25(1/2):29–53, 1998.

[63] J. Paredaens, B. Kuijpers, G. Kuper, and L. Vandeurzen. Euclid, Tarski, and Engeler Encompassed. In *Proceedings of the 6th International Workshop on Database Programming Languages*, volume 1369 of *Lecture Notes in Computer Science*, pages 1–24, Berlin, 1997. Springer-Verlag.

[64] N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, T. Özsu, B. Pernici, A. Segef, B. Theodoulidis, and U. Dayal. Towards an infrastructure for temporal databases. *SIGMOD Record*, 23(1):35–51, 1994.

[65] H. Pollard and H.G. Diamond. *The Carus Mathematical Monographs 9 (2nd edition)*, chapter The Theory of Algebraic Numbers, page 41. The Mathematical Association of America, 1975.

[66] M.F. Preparata and M.I. Shamos. *Computational Geometry.* Springer-Verlag, Berlin, 1985.

[67] P.Z. Revesz and Y. Li. MLPQ: A linear constraint database system with aggregate operators. In *Proceedings of the International Database Engineering and Applications Symposium*, pages 132–137, 1997.

[68] M. Scholl and A. Voisard, editors. *Proceedings of the 5th International Symposium on Spatial Database*, volume 1262 of *Lecture Notes in Computer Science*, Berlin, 1997. Springer-Verlag.

[69] A. Seidenberg. A new decision procedure for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

[70] P. Svensson and Z. Huang. GEO-SAL: a query language for spatial data analysis. In O. Günther and H.-J. Schek, editors, *Proceedings of the 2nd International Symposium on Spatial Database*, volume 525 of *Lecture Notes in Computer Science*, pages 119–140, Berlin, 1991. Springer-Verlag.

[71] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, Berkeley, 1951.

[72] J. D. Ullman. *Principles of Database Systems, 2nd edition.* Pitman Publishing, London, 1982.

[73] L. Vandeurzen, M. Gyssens, and D. Van Gucht. On the desirability and limitations of linear spatial query languages. In M.J. Egenhofer and J.R. Herring, editors, *Proceedings of the 4th International Symposium on Spatial Database*, volume 951 of *Lecture Notes in Computer Science*, pages 14–28, Berlin, 1995. Springer-Verlag.

[74] L. Vandeurzen, M. Gyssens, and D. Van Gucht. On query languages for linear queries definable with polynomial constraints. In E.C. Freuder, editor, *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*, pages 468–481, Berlin, 1996. Springer-Verlag.

[75] H. Whitney. Elementary structure of real algebraic varieties. *Annals of Mathematics*, 66:545–556, 1957.