QUERY LANGUAGES FOR SEMI-ALGEBRAIC DATABASES BASED ON DESCRIPTIVE COMPLEXITY OVER $\mathbb{R}$

KLAUS MEER

Abstract. We propose the study of query languages for databases involving real numbers as data (called semi-algebraic databases in the sequel). As main new aspect our approach is based on real number complexity theory as introduced in [8] and descriptive complexity for the latter developed in [17]. Using this formal framework a uniform treatment of semi-algebraic query languages is obtained. Precise results about both the data- and the expression-complexity of several such query languages are proved. More explicitly, relying on descriptive complexity theory over $\mathbb{R}$ gives the possibility to derive a hierarchy of complete languages for most of the important real number complexity classes. A clear correspondence between different logics and such complexity classes is established. In particular, it is possible to formalize queries involving in a uniform manner real spaces of different dimensions. This can be done in such a way that the logical description exactly reflects the computational complexity of a query. The latter might circumvent a problem appearing in some of the former approaches dealing with semi-algebraic databases (see [20] , [18]), where the use of first-order logic over real-closed fields can imply inefficiency as soon as the dimension of the underlying real space is not fixed - no matter whether the query under consideration is easy to compute or not.

1. Introduction

Semi-algebraic databases have raised increasing interest in recent years. Inspired by problems of computational and semi-algebraic geometry in the meanwhile there can be found many approaches dealing with real number data and real polynomial inequality constraints. To give a (by no means complete) list of references consider for example [20], [24], [19], [4], [5], [6], the survey paper [18] as well as the literature cited in there.

An important task in database theory is to study the complexity of queries asked to databases. In relation with finite model theory and descriptive complexity theory this led to numerous results on the complexity of several query languages described by logical means (see [1] for a survey). For the present work the papers [9] and [25] are of extreme interest. Here the notions of data- and expression-complexity were introduced and studied; important complexity classes in the Turing model were captured by the data- resp. expression-complexity of several such query languages. Our goal is to set up a similar theory for semi-algebraic databases. However, in order to deal in a comparable manner with data- and expression-complexity also in connection with real number data problems arise. It seems not to be clear how to treat them appropriately as long as restricting to the so far used concepts. Thus, before starting with the mathematical reasoning let us consider an elementary example from computational geometry and explain some of these problems.

Example 1.1. *Given n points in $\mathbb{R}^2$ as database the task is to select those pairs that have the largest euclidean distance. Following the approach taken in the seminal paper* [20] *(and many others as well, for example* [5],[14],[19],[24]*) this problem would be modeled as follows: the database is a finite relation $\mathfrak{R} \subset \mathbb{R}^2$ consisting of the n points. It is given as a quantifier free first-order formula over the reals having two free variables (and the formula is true exactly for the points in the database). Now in order to check whether a pair (x, y) of points realizes the largest distance with respect to pairs in $\mathfrak{R}^2$ analyze the following first-order formula over the reals with the additional symbol $\mathfrak{R}$ in the vocabulary (here $x = (x_1, x_2)$ etc.):*

$$\psi(x_1, x_2, y_1, y_2) \equiv \mathfrak{R}(x_1, x_2) \wedge \mathfrak{R}(y_1, y_2) \ \bigwedge \ \{\forall v_1, v_2, w_1, w_2 \in \mathbb{R}$$
$$(\mathfrak{R}(v_1, v_2) \wedge \mathfrak{R}(w_1, w_2)) \Rightarrow (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq (v_1 - w_1)^2 + (v_2 - w_2)^2\}$$

First of all the above problem is a real number problem; that is the point coordinates are real numbers and the complexity of the problem is depending on the quantity n of different points. Thus it is a problem perfectly suited to deal with in a corresponding model of computation. Such a model was introduced by Blum, Shub, and Smale (shortly: BSS) [8] and will be the starting point of our investigations. Second (and more important) consider the logical structure of ψ : the query is presented as a quantified first-order formula over the real closed field $\mathbb{R}$. In general, the complexity of evaluating such formulas is supposed to be tremendous since it results in quantifier elimination. In the BSS setting deciding existential first-order formulas with polynomials involved having at most degree 2 is already $NP_{\mathbb{R}}$-complete over $\mathbb{R}$ ([8]). Hence the logical shape of the above query does not reflect its complexity. Though the query is easy to compute it is expressed by a (quantified) first-order formula. In [20] this problem is circumvented by considering only databases represented by relations in a fixed space $\mathbb{R}^k$. Thus if a query on such a database is analyzed a constant bound on the quantifier depth is obtained. In such a situation the corresponding (constantly many) quantifiers can be eliminated efficiently and tractability results on the data complexity of such a query are obtained (in fact, this complexity is in NC). However, one may argue that there is no necessity to consider the above largest distance problem only in $\mathbb{R}^2$. It makes perfect sense to regard it as a single geometrical problem being posed for sets in any $\mathbb{R}^k$ and, in fact, computationally is not harder there (of course the complexity will depend on the number of points and the varying dimension k). Similarly, when analyzing the complexity of query languages over finite structures data-complexity is measured by fixing a query and evaluating it on different databases, where the latter are given as finite structures of different sizes (see [25]). In the above formulation considering the dimension k as part of the input would cause serious troubles since then the complexity of quantifier elimination enters into the consideration though the query itself remains tractable. The above problem also sheds light on a further one; the quantifiers for v_i and w_i in a particular sense are "easy" since they only range over components of points in $\mathfrak{R}$. In some of the approaches to real number databases such a difference is present when distinguishing between so-called natural and active domains (see [5],[6] as example). Here, very interesting work has been done to analyze when it is possible to replace the natural domain of a quantifier by the active one. However, even if it is possible to perform such a replacement this process once more can involve quantifier elimination and thus might destroy a possibly easy shape of a quantifier.

In the present paper we propose the use of descriptive complexity theory over the reals - as introduced in [17] - to study the complexity of query languages for semi-algebraic databases. It allows to set up precise relations between different such languages and real number complexity classes in the sense of [8]. Using meta-finite model theory [16] in a specific sense also separates "easy" from "difficult" quantifiers by allowing an exact distinction between discrete and continuous aspects of computations resp. queries. As a consequence formulas involving quantifiers which intrinsically range over an infinite domain will belong to a different logic than those incorporating bounded ones. For example the results show the (generalized) largest distance query to be describable by a first-order fixed-point formula on so-called $\mathbb{R}$-structures; such formulas only involve bounded quantifiers, thereby directly implying the largest distance problem to be computable in polynomial time (over $\mathbb{R}$).

Parallelizing the foregoing in [25] the notions of data- and expression-complexity for semi-algebraic query languages are introduced. For the main logics on (ordered) $\mathbb{R}$-structures we study these quantities and give completeness results. In section 2 we recall the basics on the BSS model of computation over $\mathbb{R}$ and on descriptive complexity theory for it. In Section 3 the notions of a semi-algebraic query, data- and expression-complexity and -completeness for a query language are introduced. Completeness results are proved in section 4.

2. Basic concepts: Complexity and descriptive complexity over $\mathbb{R}$

2.1. BSS complexity theory over the reals.

We start this section by summarizing very briefly the main ideas of real number complexity theory. For a more intensive presentation see [8],[7], and also [22] for a survey on current results in this area.

Essentially a (real) Blum-Shub-Smale (shortly: BSS) machine can be considered as a Random Access Machine over $\mathbb{R}$ which is able to perform the basic arithmetic operations at unit cost and which registers can hold arbitrary real numbers.

Definition 2.1 ([8]). a) *Let* $Y \subset \mathbb{R}^\infty := \bigoplus_{k\in\mathbb{N}} \mathbb{R}^k$, *i.e. the set of finite sequences of real numbers. A* BSS-machine M over $\mathbb{R}$ with admissible input set Y *is given by a finite set* I *of instructions labelled by* $1,\dots,N$. *A configuration of* M *is a quadruple* $(n,i,j,x) \in I\times\mathbb{N}\times\mathbb{N}\times\mathbb{R}^\infty$. *Here* n *denotes the currently executed instruction,* i *and* j *are used as addresses (copy-registers) and* x *is the actual content of the registers of* M. *The initial configuration of* M'*s computation on input* $y \in Y$ *is* $(1,1,1,y)$. *If* $n = N$ *and the actual configuration is* (N,i,j,x), *the computation stops with output* x .
The instructions M *is allowed to perform are of the following types :*

computation: : $n : x_s \leftarrow x_k \circ_n x_l$, *where* $\circ_n \in \{+,-,*,:\}$ *or* $n : x_s \leftarrow \alpha$ *for some constant* $\alpha \in \mathbb{R}$.
The register x_s *will get the value* $x_k \circ_n x_l$ *or* α *resp. All other register-entries remain unchanged. The next instruction will be* $n+1$; *moreover, the copy-register* i *is either incremented by one, replaced by* 0, *or remains unchanged. The same holds for copy-register* j.

branch: : n: **if** $x_0 \geq 0$ **goto** $\beta(n)$ **else goto** $n+1$. *According to the answer of the test the next instruction is determined (where* $\beta(n) \in I$*). All other registers are not changed.*

copy: : $n : x_i \leftarrow x_j$, *i.e. the content of the "read"-register is copied into the "write"-register. The next instruction is* $n+1$; *all other registers remain unchanged.*

b) *The* size *of an* $x \in \mathbb{R}^k$ *is* $size_{\mathbb{R}}(x) := k$. *The* cost *of any of the above operations is* 1. *The cost of a computation is the number of operations performed until the machine halts.*

c) *For some* $B \subset \mathbb{R}^\infty$ *we call a function* $f : B \to \mathbb{R}^\infty$ (BSS-) computable *iff it is realized by a BSS machine over admissible input set* B. *Similarly, a set* $A \subset B \subset \mathbb{R}^\infty$ *is* decidable *in* B *iff its characteristic function is computable.* (B,A) *is called a* decision problem.

Most of the well known boolean time-complexity classes can now be defined analogously over the reals. We here give precise definitions of three such classes and mention the others being important later on briefly.

Definition 2.2. i) *A problem* $(B,A), A \subset B \subset \mathbb{R}^\infty$ *is in class* $\mathrm{P}_{\mathbb{R}}$ *(*decidable in deterministic polynomial time*), iff there exist a polynomial* p *and a (deterministic) BSS machine* M *deciding* A *(in* B*) such that* $T_M(y) \leq p(size_{\mathbb{R}}(y))\ \forall y \in B$.

ii) (B,A) *is in* $\mathrm{NP}_{\mathbb{R}}$ *(*verifiable in non-deterministic polynomial time over $\mathbb{R}$*) iff there exist a polynomial* p *and a BSS machine* M *working on input space* $B \times \mathbb{R}^\infty$ *such that*

a) $\Phi_M(y,z) \in \{0,1\}\ \forall y \in B, z \in \mathbb{R}^\infty$
b) $\Phi_M(y,z) = 1 \implies y \in A$

c) $\forall y \in A\ \exists z \in \mathbb{R}^\infty\ \Phi_M(y,z) = 1$ *and* $T_M(y,z) \le p(size_{\mathbb{R}}(y))$

iii) (B,A) *is in* $\mathrm{DNP}_{\mathbb{R}}$ *(*digital $\mathrm{NP}_{\mathbb{R}}$*) iff* $(B,A) \in \mathrm{NP}_{\mathbb{R}}$ *but the guess space is reduced to be* $\{0,1\}^\infty$.

It is obvious how to define further classes such as $\mathrm{EXP}_{\mathbb{R}}, \mathrm{NEXP}_{\mathbb{R}}, \mathrm{DNEXP}_{\mathbb{R}}$. We will need as well some other classes which are defined similarly as their boolean counterparts. Among them are the real polynomial hierarchy $\mathrm{PH}_{\mathbb{R}}$ [10], the class $\mathrm{PAT}_{\mathbb{R}}$ of problems accepted in polynomial alternating time [10] and its digital counterpart $\mathrm{DPAT}_{\mathbb{R}}$ [11]; a set $S \subset \mathbb{R}^\infty$ is in class $\mathrm{PAT}_{\mathbb{R}}$ (resp. $\mathrm{DPAT}_{\mathbb{R}}$) iff there exist a polynomial p and a BSS machine M such that

$$S = \{x \in \mathbb{R}^\infty \,|\forall y_1 \exists z_1 \ldots \forall y_{p(|x|)} \exists z_{p(|x|)}\ M \text{ accepts } (x,y,z) \text{ in time } p(|x|)\}.$$

Here for $\mathrm{PAT}_{\mathbb{R}}$ the y_i, z_i range over $\mathbb{R}$ whereas for $\mathrm{DPAT}_{\mathbb{R}}$ they range over $\{0,1\}$.

Note that over the reals space resources alone do not provide meaningful complexity classes. Following [23] all decidable real number problems can be solved using a constant number of working registers. However, combining time and space resources makes sense [17]. We especially deal with the classes $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ and $(EXP_{\mathbb{R}}, const)$ resp. denoting those problems being decidable by a BSS machine working simultaneously both in polynomial (resp. exponential) time and constant space.

Finally, reducibility in complexity classes is defined as usual. For example, a $\mathrm{NP}_{\mathbb{R}}$-complete problem w.r.t. polynomial time reductions is given by 4-$FEAS$ (see [8]): given a (multivariate) polynomial f with real coefficients, $deg(f) \le 4$, does there exist a real zero of f?

Here we are mainly concerned with reductions in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. Our results support the suggestion in [17] to consider this class as a real counterpart of *logspace*.

2.2. **Descriptive complexity for real number computations.** Next we turn to the basic ideas of descriptive complexity over the reals and recall the notions of $\mathbb{R}$-structures and logics for them. This concept was first introduced in [17]; we refer to this paper as well as [12], [7] for a closer study of descriptive complexity in the BSS model.

We suppose the reader familiar with the main terminology of logic as well as with the concepts of vocabulary, first-order formula or sentence, interpretation and structure (see for example [15]).

Definition 2.3. *Let L_s, L_f be finite vocabularies where L_s may contain relation and function symbols, and L_f contains function symbols only. A* $\mathbb{R}$-structure of signature $\sigma = (L_s, L_f)$ *is a pair $\mathfrak{D} = (\mathfrak{A}, \mathfrak{F})$ consisting of*

i) *a finite structure $\mathfrak{A}$ of vocabulary L_s, called the* skeleton *of $\mathfrak{D}$, whose universe A will also be said to be the* universe *of $\mathfrak{D}$, and*

ii) *a finite set $\mathfrak{F}$ of functions $X : A^k \to \mathbb{R}$ interpreting the function symbols in L_f.*

Definition 2.4. *Let $\mathfrak{D}$ be a $\mathbb{R}$-structure with skeleton $\mathfrak{A}$. We denote by $|A|$ and also by $|\mathfrak{D}|$ resp. the cardinality of the universe A of $\mathfrak{A}$. This number is called the* size *of the structure $\mathfrak{D}$. A $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{F})$ is* ranked *if there is a unary function symbol $r \in L_f$ whose interpretation ρ in $\mathfrak{F}$ bijects A with $\{0, 1, \dots, |A| - 1\}$. The function ρ is called* ranking. *We will write $i < j$ for $i, j \in A$ iff $\rho(i) < \rho(j)$. A k-ranking on A is a bijection between A^k and $\{0, 1, \dots, |A|^k - 1\}$. It can easily be defined if a ranking is available. We denote by ρ^k the interpretation of the k-ranking induced by ρ.*

Throughout this paper we suppose all $\mathbb{R}$-structures to be ranked (but see Remark 3.2). We therefore notationally suppress the symbol $\le$ in the sets $\mathfrak{F}$ considered. The basic logic important for our work is first-order logic which we are going to define now.

Fix a countable set $V = \{v_0, v_1, \dots, \}$ of variables. These variables range only over the skeleton; we do not use element variables taking values in $\mathbb{R}$. This is the reason why we will be able to obtain a finer distinction of queries than just dealing with first-order logic over real closed fields.

Definition 2.5. *The language $\mathrm{FO}_{\mathbb{R}}$ contains, for each signature $\sigma = (L_s, L_f)$ a set of formulas and terms. Each term t takes, when interpreted in some $\mathbb{R}$-structure, values in either the skeleton, in which case we call it an* index term, *or in $\mathbb{R}$, in which case we call it a* number term. *Terms are defined inductively as follows*

i) *The set of index terms is the closure of the set V of variables under applications of function symbols of L_s.*
ii) *Any real number is a number term.*
iii) *If $h_1, \dots, h_k$ are index terms and X is a k-ary function symbol of L_f then $X(h_1, \dots, h_k)$ is a number term.*
iv) *If t, t' are number terms, then so are $t + t'$, $t - t'$, $t \times t'$, t/t' and $sign(t)$.*

Atomic formulas *are equalities $h_1 = h_2$ of index terms, equalities $t_1 = t_2$ and inequalities $t_1 < t_2$ of number terms, and expressions $P(h_1, \dots, h_k)$ where P is a k-ary predicate symbol in L_s and $h_1, \dots, h_k$ are index terms.*

The set of formulas of $\mathrm{FO}_{\mathbb{R}}$ is the smallest set containing all atomic formulas and which is closed under Boolean connectives and quantification $(\exists v)\psi$ and $(\forall v)\psi$. Note that we do not *consider formulas $(\exists x)\psi$ where x ranges over $\mathbb{R}$.*

Example 2.6 (cf. [17])**.** *Let L_s be the empty set and L_f be $\{r, X\}$ where both function symbols have arity 1. Then, a simple class of ranked $\mathbb{R}$-structures with signature (L_s, L_f) is obtained by letting $\mathfrak{A}$ be a finite set A, $r^{\mathfrak{D}}$ any ranking on A and $X^{\mathfrak{D}}$ any unary function $X^{\mathfrak{D}} : A \to \mathbb{R}$. Since $r^{\mathfrak{D}}$ bijects A with $\{0, 1, \dots, n - 1\}$ where $n = |A|$, this $\mathbb{R}$-structure is a point $x_{\mathfrak{D}}$ in $\mathbb{R}^{\infty}$. Conversely, for each point $x \in \mathbb{R}^{\infty}$ there is an $\mathbb{R}$-structure $\mathfrak{D}$ such that $x = x_{\mathfrak{D}}$. Thus, this class of structures models $\mathbb{R}^{\infty}$.*

On the other hand any $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{F})$ can be identified with a vector $e(\mathfrak{D}) \in \mathbb{R}^\infty$ using a natural encoding. To this aim choose a ranking on A. Without loss of generality the skeleton of $\mathfrak{D}$ can be assumed to consist of the plain set A only by replacing all functions and relations in L_s by their corresponding characteristic functions —the latter being considered as elements of the set $\mathfrak{F}$. Now using the ranking each of the functions X in $\mathfrak{F}$ can be represented by a vector $v_X \in \mathbb{R}^m$ for some appropriate m. The concatenation of all these v_X yields the encoding $e(\mathfrak{D}) \in \mathbb{R}^\infty$. Note that the length of $e(\mathfrak{D})$ is polynomially bounded in $|A|$; moreover for all $\mathbb{R}$-structures $\mathfrak{D}$, all rankings E on A and all functions $X : A^k \to \mathbb{R}$ the property that X represents the encoding $e(\mathfrak{D})$ of $\mathfrak{D}$ with respect to E is first-order expressible (see [17]*).*

Example 2.6 allows us to speak about complexity classes among $\mathbb{R}$-structures. If S is a set of $\mathbb{R}$-structures closed under isomorphisms, we say that S belongs to a complexity class $\mathfrak{C}$ over the reals if the set $\{e(\mathfrak{D}) \mid \mathfrak{D} \in S\}$ belongs to $\mathfrak{C}$.

In order to describe different complexity classes by logical means we are going to define some further logics on $\mathbb{R}$-structures.

Definition 2.7. *[*[17],[12]*]*

a) *By $FO^-_{\mathbb{R}}$ we denote the quantifier-free fragment of $FO_{\mathbb{R}}$ logic.*
b) *We say that ψ is a* second-order formula *(of signature $\sigma = (L_s, L_f)$) with free variables a ranging in some A^s if $\psi(a) = Q_1 Y_1 \dots Q_r Y_r \varphi(a)$; here φ is a first-order formula in* $\mathrm{FO}_{\mathbb{R}}$ *of signature $(L_s, L_f \cup \{Y_1, \dots, Y_r\})$ and $Q_i \in \{\exists, \forall\}$. The symbols $Y_1, \dots, Y_r$ will be called* function variables. *For a specific assignment $\bar{a}$ the sentence $\psi(\bar{a})$ is true in a $\mathbb{R}$-structure $\mathfrak{D}$ of signature σ when there exist interpretations of $Y_1, \dots, Y_r$ such that $\varphi(\bar{a})$ holds true on $\mathfrak{D}$. The set of second-order formulas will be denoted by* $\mathrm{SO}_{\mathbb{R}}$*. Together with the interpretation above it constitutes* second-order logic.
c) *If in b) quantifiers of function variables are restricted to be existential we obtain* existential second-order logic*, denoted by* $\exists\mathrm{SO}_{\mathbb{R}}$.
d) *If in an existential second-order formula all quantified function variables are supposed to have an image in $\{0, 1\}$ we get* existential digital second-order logic $\exists D\mathrm{SO}_{\mathbb{R}}$.

Finally, we need to introduce fixed point logic for $\mathbb{R}$-structures. This is done by combining a maximization with a fixed-point rule.

Let $F(s, \bar{t})$ be a number term with free variables s and $\bar{t} = (t_1, \dots, t_r)$. Then $\max_s F(s, \bar{t})$ is also a number term with free variables $\bar{t}$. Its interpretation in any $\mathbb{R}$-structure $\mathfrak{D}$ and for any point $u \in A^r$ interpreting $\bar{t}$ is the maximum of $F^{\mathfrak{D}}(a, u)$ where a ranges over A.

Next, fix a signature $\sigma = (L_s, L_f)$, an integer $r \geq 1$, and a pair (Z, D) of function symbols both of arity r and not contained in this signature. Let $F(Z, \bar{t})$ and $H(D, \bar{t})$ be number terms of signature $(L_s, L_f \cup \{Z, D\})$ and free variables $\bar{t} = (t_1, \dots, t_r)$. Note that Z can appear several times in F and we do not require that its arguments are $t_1, \dots, t_r$. The only restriction is that the number of free variables in F coincides with the arity of Z. A similar remark holds for H and D.

For any $\mathbb{R}$-structure $\mathfrak{D}$ of signature σ and any interpretation $\zeta : A^r \to \mathbb{R}$ of Z and $\Delta : A^r \to \mathbb{R}$ of D respectively the number terms $F(Z, \bar{t})$ and $H(D, \bar{t})$ define functions

$$F_\zeta^{\mathfrak{D}}, H_\Delta^{\mathfrak{D}} : A^r \to \mathbb{R}.$$

Let us consider the sequence of pairs $\{\Delta^i, \zeta^i\}_{i \geq 0}$ with $\zeta^i : A^r \to \mathbb{R}$ and $\Delta^i : A^r \to \mathbb{R}$ inductively defined by

$$\begin{aligned} \Delta^0(x) = 0 & \qquad \text{for all } x \in A^r \\ \zeta^0(x) = 0 & \qquad \text{for all } x \in A^r \\ \Delta^{i+1}(x) \quad = & \quad \begin{cases} H_{\Delta^i}^{\mathfrak{D}}(x) & \text{if } \Delta^i(x) = 0 \\ \Delta^i(x) & \text{otherwise} \end{cases} \\ \zeta^{i+1}(x) \quad = & \quad \begin{cases} F_{\zeta^i}^{\mathfrak{D}}(x) & \text{if } \Delta^i(x) = 0 \\ \zeta^i(x) & \text{otherwise.} \end{cases} \end{aligned}$$

Since $\Delta^{i+1}(x)$ only differs from $\Delta^i(x)$ in case the latter is zero, one has that $\Delta^j = \Delta^{j+1}$ for some $j < |A|^r$. In this case, moreover, we also have that $\zeta^j = \zeta^{j+1}$. We denote these fixed points by Z^∞ and D^∞ and call them the *fixed points of* $F(Z, \bar{t})$ *and* $H(D, \bar{t})$ *on* $\mathfrak{D}$. We say that $F^{\mathfrak{D}}$ *updates* ζ.

Note that D plays the role of the characteristic function for the domain of Z and the different Δ^i determine the successive updatings of this domain. We say that Z^∞ *is defined on* x if $D^\infty(x) \neq 0$.

The fixed point rule is now stated in the following way. If $F(Z, \bar{t})$ and $H(D, \bar{t})$ are number terms as above then

$$\mathbf{fp}[Z(\bar{t}) \leftarrow F(Z, \bar{t}), H(D, \bar{t})](\bar{u})$$

and

$$\mathbf{fp}[D(\bar{t}) \leftarrow H(D, \bar{t})](\bar{u})$$

are number terms of signature (L_s, L_f). Their interpretations on a given $\mathbb{R}$-structure $\mathfrak{D}$ are $Z^\infty(\bar{u})$ and $D^\infty(\bar{u})$ respectively.

Definition 2.8. *Fixed point logic* for $\mathbb{R}$-structures, denoted by $\mathrm{FP}_\mathbb{R}$, is obtained by augmenting first-order logic $\mathrm{FO}_\mathbb{R}$ with the maximization rule and the fixed point rule.

The main results in [17] and [12] show that the logics introduced above are closely related to real number complexity classes in the following sense:

Theorem 2.9 ([17],[12])**.** *Let (F, F^+) be a decision problem of (ranked) $\mathbb{R}$-structures as explained in Example 2.6. Then (F, F^+) belongs to complexity class $\mathfrak{C}$ if and only if there exists an formula ψ in language $\mathfrak{L}$ such that $F^+ = \{\mathfrak{D} \in F | \mathfrak{D} \models$*

$\psi\}$. Here for $\mathfrak{C}$ and $\mathfrak{L}$ one can substitute one of the pairs $(\mathrm{P}_{\mathbb{R}}, \mathrm{FP}_{\mathbb{R}})$, $(\mathrm{DNP}_{\mathbb{R}}, \exists\mathrm{DSO}_{\mathbb{R}}), (\mathrm{NP}_{\mathbb{R}}, \exists\mathrm{SO}_{\mathbb{R}})$, *and* $(\mathrm{PH}_{\mathbb{R}}, \mathrm{SO}_{\mathbb{R}})$. ■

3. Semi-algebraic databases and queries; data- and expression complexity

Once having introduced descriptive complexity over $\mathbb{R}$ it is straightforward how to base the notion of semi-algebraic databases and semi-algebraic queries on it. We closely follow the approaches in [9],[25],[1] by performing the changes necessary when working over the reals.

Definition 3.1. *Fix a signature $\sigma = (L_s, L_f)$ of finite vocabularies L_s and L_f where L_s may contain relation and function symbols and L_f contains function symbols only.*

a) *A* semi-algebraic database *of type σ is a (ranked) $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{F})$ of signature σ.*
b) *Fix a natural number k. A* semi-algebraic query *Q of type (σ, k) is a partial map assigning (if defined) to every semi-algebraic database $\mathfrak{D}$ of type σ a relation of arity k defined on the finite universe of $\mathfrak{D}$.*
c) *A semi-algebraic query is* computable *if it is BSS-computable with respect to the usual encoding of $\mathbb{R}$-structures as inputs for BSS machines* [17].

Remark 3.2. a) *Semi-algebraic queries could be defined more generally by mapping $\mathfrak{D}$ to another database of (possibly) different signature. The expositions in* [1] *and in* [25] *for such a restriction mutis mutandis in our setting hold as well.*
b) *Just as in descriptive complexity theory over finite structures the problem of genericity in our setting appears as well (see* [17]*). In principle one can think about similar ways to deal with it in connection with $\mathbb{R}$-structures such as defining generic queries (cf.* [1]*) or extending the definition of special devices such as "generic machines"* [2] *or reflectional relational machines* [3] *to the BSS model. Since we want to focus on outlining our approach for semi-algebraic databases throughout this paper we assume all $\mathbb{R}$-structures to be ranked.*

In this paper we are considering semi-algebraic query languages arising from different logics on $\mathbb{R}$-structures. Given a formula $\rho(a_1, \ldots, a_s)$ in a particular logic for $\mathbb{R}$-structures of signature σ and with free variables $a_1, \ldots, a_s$ ranging over the finite universe, the corresponding query Q_ρ is defined by assigning to an $\mathbb{R}$-structure $\mathfrak{D}$ of signature σ that relation of arity s determined by all assignments for the a_i making ρ a true sentence in $\mathfrak{D}$.

Definition 3.3 (cf. [25])**.** a) *Given a semi-algebraic query Q_ρ of signature (σ, k) and corresponding to a formula ρ in the above sense its* graph *is the set $Gr(Q_\rho) := \{(a, \mathfrak{D}) | a \in A^k, \mathfrak{D} \models \rho(a)\}$.*

b) *Given a semi-algebraic database* $\mathfrak{D}$ *of signature* σ *its* graph *with respect to a language* L *is the set* $Gr_L(\mathfrak{D}) := \{(a, \rho) | \rho$ *is a formula in* L *of arity* $s, a \in A^s, \mathfrak{D} \models \rho(a)\}$.

We are interested in the BSS complexity of deciding membership in one of the two graph sets defined above. This means either the language is fixed and we measure the complexity for varying semi-algebraic databases as function in the size of the database; or vice versa the database is fixed and the complexity for varying formulas in a given language is studied as function in the formula length. In the following the notion of completeness of particular languages refers to reduction algorithms running simultaneously in polynomial real time and constant space. Our results substantiate that the according class $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ can be seen as an analogue of *logspace* in the BSS setting.

Definition 3.4 (cf. [25])**.** a) *The* data-complexity *of a language* L *for semi-algebraic databases is the BSS complexity of deciding membership of* $(a, \mathfrak{D})$ *in* $Gr(Q_\rho)$ *for any fixed* $\rho \in L$. *A language* L *is* data-complete *for a real number complexity class* $\mathfrak{C}$ *if* $\forall \rho \in L$ *deciding membership in* $Gr(Q_\rho)$ *is in* $\mathfrak{C}$ *and furthermore there exists a* $\rho_0 \in L$ *such that the latter problem for the graph* $Gr(Q_{\rho_0})$ *is complete in* $\mathfrak{C}$ *with respect to reductions in* $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$.

b) *The* expression-complexity *of a language* L *for semi-algebraic databases is the BSS complexity of deciding membership of* (a, ρ) *in* $Gr_L(\mathfrak{D})$ *for any fixed* $\mathbb{R}$*-structure* $\mathfrak{D}$ *and* $\rho \in L$ *is a formula over the corresponding signature. A language* L *is* expression-complete *for a real number complexity class* $\mathfrak{C}$ *if for any fixed semi-algebraic database* $\mathfrak{D}$ *the graph membership problem is in* $\mathfrak{C}$ *and furthermore there exists a semi-algebraic database* $\mathfrak{D}_0$ *such that the latter problem with respect to* $\mathfrak{D}_0$ *is complete in* $\mathfrak{C}$ *under reductions in* $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. *Here complexity is measured in the* length *of a given formula* ρ. *The latter intuitively is the number of symbols necessary to write down* ρ*, where quantifiers, symbols from the signature, logical and real arithmetic operation symbols and - most important - real numbers are supposed to have unit size. We skip a formal definition which is easily done by induction.*

In the following we will notationally identify a language for semi-algebraic databases and the semi-algebraic queries resulting from it.

4. Completeness results

For most of the logics introduced in [17] and [12] completeness results for both data- and expression-complexity are shown now. The results especially settle a precise correspondence between the logical structure of a semi-algebraic query and its complexity. This removes the difficulties arising when formalizing easy queries already with first order logic for real closed fields.

Theorem 4.1. *The language* $\mathrm{FO}_{\mathbb{R}}^-$ *of first order logic without quantifiers is both data-complete and expression-complete in* $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$.

Proof. Since we consider reductions in class $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ for both statements it suffices to show membership of the corresponding graph problem in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. Let a formula $\rho \in \mathrm{FO}_{\mathbb{R}}^{-}$ with s free variables be fixed. Consider a semi-algebraic database $\mathfrak{D} = (\mathfrak{A}, F)$ of size n and an $a \in A^s$ as input. Evaluating ρ in a can be done in a number of steps proportional to the length of ρ, which is independent of n. Hence there exists a BSS algorithm working in polynomial (even constant) time and constant space.

Now let the semi-algebraic database be fixed and consider as input an $a \in A^s$ together with $\rho \in \mathrm{FO}_{\mathbb{R}}^{-}$ of length n. In order to evaluate ρ in a we can proceed as in [21] for evaluating Boolean expressions. However, the corresponding algorithm has to be performed simultaneously in polynomial time and constant space. (Note that Michaux' result [23] guarantees this task to be executable in constant space, but his algorithm in general might result in a super-polynomial slow-down). Nevertheless, a closer analysis of Lynch's method shows that in this particular situation Michaux' foregoing will not result in a slow-down: The basic ingredient of Lynch's method is the organization of how to cycle through the sub-formulas of ρ. To this aim in the Turing model at most $O(\log n)$ many columns are introduced to record specific sub-results (i.e. values in $\{0, 1\}$) and to indicate which part is going to be evaluated next. Hence, we get at most $O(\log n)$ integers of bounded value (independently of n) to store. This means that the entire information can be stored in a single integer of magnitude at most 2^{2^n} (as a very crude upper bound). The key point with respect to Michaux' coding argument now is that there is only one single integer $M \le 2^{2^n}$ holding all the necessary information. In this particular situation his method works without super-polynomial time trade-off; the information in M can be decoded and encoded in polynomial time (using repeated squaring). Thus, an adaption of Lynch's algorithm for $\mathrm{FO}_{\mathbb{R}}^{-}$ can be performed in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. □

Next we consider full first order logic on $\mathbb{R}$-structures.

Theorem 4.2. *The language* $\mathrm{FO}_{\mathbb{R}}$ *of first order logic is data-complete in* $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. *It is expression-complete in* $\mathrm{DPAT}_{\mathbb{R}}$*, the class of problems decidable in polynomial digital alternating time over* $\mathbb{R}$ *(*[11]*).*

Proof. Concerning data-complexity we again only have to show containment of the graph membership problem in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. Since all semi-algebraic databases considered here by convention are ordered one can cycle through all possible tuples of assignments of those variables quantified in the given formula

$$\rho(a) \equiv Q_1 x_1 Q_2 x_2 \dots Q_k x_k \; \psi(x_1, \dots , x_k, a) \;, \; Q_i \in \{\forall, \exists\} \text{ and } \psi \in \mathrm{FO}_{\mathbb{R}}^{-}.$$

Since the number of quantifiers in ρ is fixed and since $\mathrm{FO}_{\mathbb{R}}$ quantifiers only range over the finite universe there are polynomially (namely n^k) many different assignments for the x_i only (where n denotes the size of the database). For each of them $\psi(x, a)$ can be evaluated in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ according to the last theorem. It remains to plug together the polynomially many truth values and to check whether they satisfy the quantifier prefix. Again an argument on the magnitude of integers coding all this information yields the claim: Given the quantifier prefix consider all those subsets T of assignments for the x_i's with the following property: $\rho(a)$ is evaluated "true" if exactly for all $x \in T$ the sentence $\psi(x,a)$ is true. Such a T can be coded by an integer $\leq 2^{n^k}$ and all such subsets can be represented via a single integer $\leq 2^{2^{n^k}}$. Since integers of such magnitude are computable in time $O(n^k)$ in the BSS model, all the information about the truth values of ψ necessary to make ρ true can be coded in polynomial time in a single register. As above, it follows that for given $(\mathfrak{D}, a)$ $\mathfrak{D} \models \rho(a)$ can be checked in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$.

Next, consider a semi-algebraic database $\mathfrak{D}$ to be given. We want to show that deciding membership in $Gr_{\mathrm{FO}_{\mathbb{R}}}(\mathfrak{D})$ is in class $\mathrm{DPAT}_{\mathbb{R}}$. To this aim we have to show (see [11]) the existence of a polynomial p and a BSS machine M such that

$$\begin{array}{rl}(a,\rho) \in Gr_{\mathrm{FO}_{\mathbb{R}}}(\mathfrak{D}) \iff & \exists y_1 \forall z_1 \exists y_2 \forall z_2 \ldots \exists y_{p(|\rho|)} \forall z_{p(|\rho|)}\ M \text{ accepts the input} \\ & (a, \rho, y_1, z_1, \ldots, y_{p(|\rho|)}, z_{p(|\rho|)}) \text{ in time } \leq p(|\rho|)\end{array}.$$

W.l.o.g. we assume the universe of $\mathfrak{D}$ to be $A := \{0,1\}$. Let a formula $\rho(a) \equiv Q_1 x_1 \ldots Q_k x_k \psi(a,x) \in \mathrm{FO}_{\mathbb{R}}$ of length n be given; clearly, $k \leq n$. Thus we can choose a linear polynomial p in the definition of $\mathrm{DPAT}_{\mathbb{R}}$ satisfying the following: For any formula ρ of length n the machine M on input (a,ρ) first matches the quantifier shape $Q_1 \ldots Q_k$ with a corresponding shape in $\exists y_1 \ldots \forall z_{p(|\rho|)}$ and then evaluates ψ on $(x_1, \ldots, x_k, a, \rho)$ in linear time.

Let us turn to completeness. The following problem $QBSCP$ (quantified binary satisfiability circuit problem) is complete under $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ reduction in class $\mathrm{DPAT}_{\mathbb{R}}$: given an algebraic circuit $\mathfrak{B}$ with n inputs and a quantifier prefix $Q_1 \ldots Q_n \in \{\exists, \forall\}^n$, decide whether $Q_1 x_1 \ldots Q_n x_n\ \mathfrak{B}$ accepts $(x_1, \ldots, x_n)$. Here the x_i range over $\{0,1\}$. Its completeness can be easily established in a similar manner the existence of a $\mathrm{DNP}_{\mathbb{R}}$-complete problem is shown in [11]. Just note that the Lagrange polynomials used therein can be evaluated in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. $QBSCP$ can be obtained as a membership problem for a particular $Gr_{\mathrm{FO}_{\mathbb{R}}}(\mathfrak{D}_0)$. Choose a semi-algebraic database $\mathfrak{D}_0 := (\mathfrak{A}_0, \mathfrak{F}_0)$ where the universe is $A_0 = \{0,1\}$, the finite structure associated with $\mathfrak{A}_0$ is empty and $\mathfrak{F}_0$ consists of the constant nullary function $1 \in \mathbb{R}$ only. Any $OBSCP$ instance can then be expressed as a membership task for $Gr_{\mathrm{FO}_{\mathbb{R}}}(\mathfrak{D}_0)$ by representing the given algebraic circuit by a $\mathrm{FO}_{\mathbb{R}}^{-}$ formula of polynomially bounded length. $\square$

As shown in [17] (see Theorem 2.9) over ordered $\mathbb{R}$-structures first order fixed-point logic captures $\mathrm{P}_{\mathbb{R}}$. This implies

Theorem 4.3. *The language* $\mathrm{FP}_{\mathbb{R}}$ *of first order fixed-point logic is data-complete in* $\mathrm{P}_{\mathbb{R}}$ *and expression-complete in* $\mathrm{EXP}_{\mathbb{R}}$.

Proof. For data-completeness membership in $\mathrm{P}_{\mathbb{R}}$ is a consequence from [17]. Completeness follows by reduction from the problem $SUBS$ of deciding, whether a system of quadratic polynomial equations and inequalities over $\mathbb{R}$ is solvable by substitution (see [13]). The PPT reduction algorithm used in [13] in order to establish $\mathrm{P}_{\mathbb{R}}$-completeness of $SUBS$ can be performed in $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$ as well: the time used by each processor in those reductions is constant.

Concerning expression-complexity membership in $\mathrm{EXP}_{\mathbb{R}}$ is immediate by combining the arguments in the proofs of Theorem 4.1 and 4.2. Cycling through all possible assignments of the quantified variables works in time $\mathrm{EXP}_{\mathbb{R}}$ and the same holds true for evaluating a fixed-point construct. Completeness is established by reducing any $\mathrm{EXP}_{\mathbb{R}}$ problem to the membership problem for $Gr_{\mathrm{FP}_{\mathbb{R}}}(\mathfrak{D}_0)$ where $\mathfrak{D}_0$ is as in the proof of Theorem 4.2. This works exactly as in [17] describing the actions of an $\mathrm{EXP}_{\mathbb{R}}$ machine M by a function $Z : A^m \to \mathbb{R}$ for a suitable $m \in \mathbb{N}$ being linear in the size of the current input for M. The steps performed by M as usually are simulated using the fixed-point rule. Since for a given problem in $\mathrm{EXP}_{\mathbb{R}}$ M is fixed this reduction works within complexity class $(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$. $\square$

Consider once more Example 1.1. In our approach the problem is represented by a semi-algebraic database $\mathfrak{D} = (\mathfrak{A}, \mathfrak{F})$ where $A = \{1, \dots, n\}$ and $\mathfrak{F}$ consists of one function symbol $X : A^2 \to \mathbb{R}$; here $X(\bullet, j)$ for fixed j represents the j-th point of the database $\mathfrak{R}$ (for simplicity we have chosen here the dimension $\mathbb{R}^n$ equal to the number of points. This can be generalized without difficulty). Now the largest distance problem is expressed by a $\mathrm{FP}_{\mathbb{R}}$ formula: for two points $X(\bullet, i)$ and $X(\bullet, j) \in \mathbb{R}^n$ in $\mathfrak{R}$ (i.e. $i, j \in A$ and $X(k, i)$ is the k-th component of $X(\bullet, i)$) define a function which sums up the n components $(X(k, i) - X(k, j))^2, 1 \le k \le n$. The latter function is describable as a fixpoint. Thus the related query can be seen to be computable in polynomial time according to Theorem 2.9 just by analyzing its logical form. No quantifier elimination over a real closed field is involved in the logical description.

We next turn to logics involving second order constructs. This means one can quantify over functions from the finite universe to $\mathbb{R}$. Thus second order logic is related to quantifier elimination problems for real closed fields and therefore to nondeterminism as defined in the BSS model.

Theorem 4.4. *The language $\exists\mathrm{SO}_{\mathbb{R}}$ of existential second order logic is data-complete in $\mathrm{NP}_{\mathbb{R}}$ and expression-complete in $\mathrm{NEXP}_{\mathbb{R}}$.*

Proof. Containment in the corresponding complexity classes follows from [17] by guessing the existentially quantified functions and evaluating the remaining $\mathrm{FO}_{\mathbb{R}}$ formula. Completeness is shown as in Theorem 4.3. $\square$

Without proof we mention the digital version of existential second order logic to capture the digital analogues of $\mathrm{NP}_{\mathbb{R}}$ and $\mathrm{NEXP}_{\mathbb{R}}$.

Corollary 4.5. *The language $\exists\mathrm{DSO}_{\mathbb{R}}$ of digital existential second order logic is data-complete in $\mathrm{DNP}_{\mathbb{R}}$ and expression-complete in $\mathrm{DNEXP}_{\mathbb{R}}$.* $\blacksquare$

Finally, we treat $\mathrm{SO}_{\mathbb{R}}$ logic.

Theorem 4.6. *The language* $\mathrm{SO}_{\mathbb{R}}$ *of second order logic is data-complete in* $\mathrm{PH}_{\mathbb{R}}$ *and expression-complete in* $\mathrm{PAT}_{\mathbb{R}}$*, the class of problems decidable in polynomial alternating time over* $\mathbb{R}$*.*

Proof. This is almost the same as that for Theorem 4.2. Now the second order quantifiers range over elements in $\mathbb{R}$ (more precisely: over function variables), the reason why on the data-complexity side one jumps to the polynomial hierarchy. For expression-complexity it yields in replacing the binary quantifiers by "full" ones, i.e. switching from $\mathrm{DPAT}_{\mathbb{R}}$ to $\mathrm{PAT}_{\mathbb{R}}$. □

We have seen that using descriptive complexity theory over the real numbers as ground tool for dealing with queries to semi-algebraic databases allows a uniform treatment both of data- and expression-complexity issues. A strong correspondence between the logic on $\mathbb{R}$-structures necessary to describe a query and the complexity of computing the latter can be settled. Especially it is possible to treat queries for semi-algebraic databases whose data belong to real euclidean spaces of different dimension. The following table summarizes our results:

Logic	data-complete in	expression-complete in
$\mathrm{FO}_{\mathbb{R}}^{-}$	$(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$	$(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$
$\mathrm{FO}_{\mathbb{R}}$	$(\mathrm{P}_{\mathbb{R}}, \mathrm{const})$	$\mathrm{DPAT}_{\mathbb{R}}$
$\mathrm{FP}_{\mathbb{R}}$	$\mathrm{P}_{\mathbb{R}}$	$\mathrm{EXP}_{\mathbb{R}}$
$\exists\mathrm{DSO}_{\mathbb{R}}$	$\mathrm{DNP}_{\mathbb{R}}$	$\mathrm{DNEXP}_{\mathbb{R}}$
$\exists\mathrm{SO}_{\mathbb{R}}$	$\mathrm{NP}_{\mathbb{R}}$	$\mathrm{NEXP}_{\mathbb{R}}$
$\mathrm{SO}_{\mathbb{R}}$	$\mathrm{PH}_{\mathbb{R}}$	$\mathrm{PAT}_{\mathbb{R}}$

Similar results can be obtained for the other logics considered in [17] and [12].

We would like to stress that the basic concept of k-ary generalized tuples introduced in [20] is captured by our foregoing as well. If in the former setting a geometric object is given by a finite collection of quantifier free formulas, in the latter we can easily describe these formulas (resp. the polynomials involved) as a $\mathbb{R}$-structure (see [17]). Moreover, the size of this $\mathbb{R}$-structure is determined by the important quantities influencing complexity issues, namely the number of variables (which has not to be fixed), the degree of the polynomials in the describing formula and the number of constraints.

An important area for further research in our opinion is the closer study of unranked $\mathbb{R}$-structures, combined with the above mentioned idea to deal with genericity. Here we believe an extension of the ideas developed in [2] and [3] to be promising.

5. Acknowledgement

This work was done while the author was staying at the Mathematical Sciences Research Institute, Berkeley, US. The stay was made possible by partial support both from the MSRI and the Deutsche Forschungsgemeinschaft, AZ: Me 1424/3-1. Thanks go to C. Michaux for helpful discussions.

References

[1] S. Abiteboul, R. Hull, V. Vianu: Foundations of Databases. Addison-Wesley (1995).

[2] S. Abiteboul, V. Vianu: Generic Computation and its Complexity. Proc. 23rd STOC (1991), 209–219.

[3] S. Abiteboul, C.H. Papadimitriou, V. Vianu: The Power of Reflectional Relational Machines. Proc. LICS (1994).

[4] M. Benedikt, G. Dong, L. Libkin, L. Wong: Relational Expressive Power of Constraint Query Languages. Journal of the ACM 45, No.1 (1998), 1–34.

[5] M. Benedikt, L. Libkin: On the Structure of Queries in Constraint Query Languages, Proc. LICS 1996, 25–34.

[6] M. Benedikt, L. Libkin: Languages for Relational Databases over Interpreted Structures, Proc. PODS 1997, 87–98.

[7] L. Blum, F. Cucker, M. Shub, S. Smale: Complexity and Real Computation. Springer-Verlag (1998).

[8] L. Blum, M. Shub, S. Smale: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bulletin American Mathematical Society **21** (1989), 1–46.

[9] A.K. Chandra, D. Harel: Computable Queries for Relational Data Bases. Journal of Computer and System Sciences 21 (1980), 156–178.

[10] F. Cucker: On the complexity of quantifier elimination: the structural approach. The Computer Journal 36 (1993), 400–408.

[11] F. Cucker and M. Matamala: On digital nondeterminism. Mathematical Systems Theory 29 (1996), 635–647.

[12] F. Cucker, K. Meer: Logics which capture complexity classes over the reals. To appear in: Journal of Symbolic Logic.

[13] F. Cucker, A. Torrecillas: Two P-complete Problems in the Theory of the Reals. Journal of Complexity 8 (1992), 454–466.

[14] F. Dumortier, M. Gyssens, L. Vandeurzen, D. van Gucht: On the decidability of Semi-Linearity for Semi-Algebraic Sets and its Implications for Spatial Databases. Proc. PODS (1997), 68–77.

[15] H.D. Ebbinghaus, J. Flum: Finite Model Theory. Springer-Verlag (1995).

[16] E. Grädel and Y. Gurevich. Metafinite model theory. In: D. Leivant (ed.), Logic and Computational Complexity. Springer-Verlag (1996), 313–366.

[17] E. Grädel, K. Meer: Descriptive complexity theory over the real numbers. Proc. STOC (1995) 315–324. Full paper version in: J. Renegar, M. Shub, and S. Smale (editors): The Mathematics of Numerical Analysis, Lectures in Applied Mathematics **32**, American Mathematical Society (1996), 381–404.

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

[19] M. Gyssens, J. van den Bussche, D. van Gucht: Complete geometrical query languages. Proc. PODS (1997), 62–67.

[20] P.C. Kanellakis, G.M. Kuper, P.Z. Revesz: Constraint Query Languages, Journal of Computer and System Sciences 51 (1995), 26–52.

[21] N. Lynch: Logspace recognition and translation of parenthesis languages. Journal of the ACM 24 (1977), 583–590.

[22] K. Meer, C. Michaux: A Survey on Real Structural Complexity Theory. Bulletin of the Belgian Mathematical Society Simon Stevin 4 (1997), 113–148.

[23] C. Michaux: Une remarque à propos des machines sur $\mathbb{R}$ introduites par Blum, Shub et Smale. C. R. Acad. Sci. Paris, 309, Série I (1989), 435–437.

[24] J. Paredaens, J. Van den Bussche, D. van Gucht: First-order queries on finite structures over the reals. SIAM Journal on Computing 27, no.6 (1998), 1747–1763.

[25] M. Vardi: Complexity of relational query languages. Proc. 14th STOC (1982), 137–146.

Klaus Meer, Lehrstuhl C fuer Mathematik, RWTH Aachen, Templergraben 55, D-52062 Aachen, Germany

E-mail address: `meer@rwth-aachen.de`