Descriptive Complexity Theory over the Real Numbers*

Erich Grädel† Klaus Meer‡

RWTH Aachen

Abstract

We present a logical approach to complexity over the real numbers with respect to the model of Blum, Shub and Smale.

The logics under consideration are interpreted over a special class of two-sorted structures, called $\mathbb{R}$-*structures*: They consist of a finite structure together with the ordered field of reals and a finite set of functions from the finite structure into $\mathbb{R}$. They are a special case of the *metafinite structures* introduced recently by Grädel and Gurevich. We argue that $\mathbb{R}$-structures provide the right class of structures to develop a descriptive complexity theory over $\mathbb{R}$. We substantiate this claim by a number of results that relate logical definability on $\mathbb{R}$-structures with complexity of computations of BSS-machines.

1 Introduction

Computations with real numbers. In 1989 Blum, Shub, and Smale [2] introduced a model for computations over the real numbers (and other rings as well) which is now usually called a BSS machine. The important difference with, say, the Turing model is that real numbers are treated as basic entities and that arithmetic operations on the reals are performed in a single step, independently of the magnitude or complexity of the involved numbers. In particular, the model abstracts from the problems that in actual computers real numbers have to be approximated by bit-sequences, that the complexity of arithmetic operations depends on the length of these approximate representations, from rounding errors and from the problem that exact testing for 0 is impossible in practice. Similar notions of computations over arbitrary fields or rings had been investigated earlier in *algebraic complexity theory* (see [22, 27, 29] for survey articles and the forthcoming book [3] for a comprehensive treatment). A novelty of the approach of Blum, Shub and Smale

*A preliminary version of this paper has been presented at the ACM Symposium on Theory of Computing, Las Vegas, May 1995.

†Lehrgebiet Mathematische Grundlagen der Informatik, RWTH Aachen, D-52056 Aachen, graedel@informatik.rwth-aachen.de

‡Lehrstuhl C für Mathematik, RWTH Aachen, D-52056 Aachen, meer@rwth-aachen.de. Partially supported by Deutsche Forschungsgemeinschaft DFG under contract Me 1424/2-1 and by EC-Project NeuroCOLT.

is that their model is uniform (for all input-lengths) whereas the notions explored in algebraic complexity (straight-line programs, arithmetic circuits, decision trees) are typically non-uniform. One of the main purposes of the BSS approach was to create a uniform complexity theory dealing with problems having an analytical and topological background, and to show that certain problems remain hard even if arbitrary reals are treated as basic entities.

Many basic concepts and fundamental results of classical computability and complexity theory reappear in the BSS model: the existence of universal machines, the classes $P_{\mathbb{R}}$ and $NP_{\mathbb{R}}$ (real analogues of P and NP) and the existence of $NP_{\mathbb{R}}$-complete problems. Of course these notions appear in a different form, with a strong analytic flavour: typical examples of undecidable, recursively enumerable sets are complements of certain Julia sets, and the first problem that was shown to be $NP_{\mathbb{R}}$-complete is the question whether a given multivariate polynomial of degree four has a real root [2]. Like in the Boolean setting all problems in the class $NP_{\mathbb{R}}$ are decidable within single exponential time (but this is not as trivial as in the classical case) and the $P_{\mathbb{R}}$ versus $NP_{\mathbb{R}}$ question is one of the major open problems .

However, there also are many differences between Boolean and real complexity theory. Just to mention a few we note that the meaning of space resources seems to be very different (cf. [25] and section 6 below), that certain separation results can be established (like $NC_{\mathbb{R}} \subsetneq P_{\mathbb{R}}$ and $NP_{\mathbb{R}} \subsetneq EXP_{\mathbb{R}}$ [4]) which are open in the Boolean theory, and that some discrete problems seem to change their complexity behaviour when considered in the BSS model (cf. [23]). Nevertheless even from the discrete point of view the continuous approach provides interesting results; in particular, Boolean parts of real complexity classes bear a strong relationship to the classical theory (see [5, 21]).

For a more detailed treatment we refer the interested reader to the forthcoming book [1] as well as the survey paper [24] and the references given there.

Logics that capture complexity classes. While computational complexity theory is concerned with the amount of resources like time or space that are neccessary to solve a problem, *descriptive complexity theory* is a branch of finite model theory, that considers the logical complexity of defining a property or query, and relates computational complexity with logical definability.

One of the most important results is Fagin's theorem [10] which provides a purely logical, machine-independent characterization of NP: Let L be an isomorphism-closed class of finite structures of a fixed, nonempty signature. Then L is in NP if and only if there exists an existential second-order sentence ψ such that L is precisely the class of finite models of ψ. Some years later, Immerman and Vardi [17, 28] proved that, on ordered structures, the problems solvable in polynomial time are exactly those definable in least fixed-point logic. Immerman [17]–[20] systematically studied the problem of capturing complexity classes by logical languages and came up with logical characterizations for most of the major complexity classes. The most important results in this field are surveyed in [16, 20]; for a textbook on finite model theory we refer to [9].

These logical characterizations of complexity are model theoretic and based either on fragments of second-order logic [10, 12] or on extensions of first-order logic by additional means to construct new relations (such as fixed point operators). There also is a different, functional approach which draws more on recursion theory than model theory. Gurevich proved that interpreting the classical calculus of primitive recursive functions (resp. recursive functions) over finite successor structures gives precisely the log-space (resp. polynomial time) computable global functions [15, 16]. A variant of the latter result was also obtained by Sazonov [26].

It should be noted, that for complexity classes below NP, precise logical characterizations have been obtained only on ordered structures. The problem whether there exists a logic capturing precisely the polynomial-time computable queries on arbitrary finite structures is still open.

Metafinite models and a descriptive complexity theory over $\mathbb{R}$. In many areas of computer science (e.g. databases, optimization, fault-tolerant networks) there arise objects that consist of a finite structure together with a collection of numbers from an infinite numerical domain, on which arithmetical operations are performed. To reason about such objects in an adequate way, Grädel and Gurevich have recently introduced *metafinite structures* [14]. Metafinite model theory should be viewed as a possibility for going beyond finite models, in order to integrate logical queries to finite structures with arithmetical operations on numbers from an infinite structure like, e.g., the ring of integers or the ordered field of reals. To avoid problems pertinent to typical infinite numerical structures — e.g. the undecidability of arithmetic — Grädel and Gurevich have investigated a number of logics for metafinite models that extend the usual logics on finite structures by *restricted access* to the infinite numerical domain. The relationship of metafinite model theory with other proposals (most notably from database theory) for combining finite structures with infinite numerical domains is also discussed in [14].

In this paper we exploit metafinite models to develop a logical approach to complexity over $\mathbb{R}$ for the BSS model, and obtain results that parallel those of descriptive complexity theory in the Boolean case. Indeed, the first problem for such an approach is to define the right class of structures to model in a convincing way the computational problems that we are interested in, and to admit logical reasoning about computations of BSS machines within natural logical languages. Since we are dealing with the ordered field of reals, we are not in the realm of finite model theory anymore. On the other side, the model theory of real closed fields, or of the ring $\mathbb{R}^\infty$ does not provide the right framework either. What is needed is a clear separation between the finite, discrete aspects of problems and computations (like indices of tuples, time, indices of registers, the finite control of the machines) on one side and the arithmetic of real numbers on the other side.

This can be achieved by $\mathbb{R}$-structures, which are a special case of metafinite models. They consist of a finite structure $\mathfrak{A}$, together with the ordered field of reals and a finite set of functions from $\mathfrak{A}$ into $\mathbb{R}$. We discuss a number of logical languages on $\mathbb{R}$-structures and investigate their expressive power. We argue that $\mathbb{R}$-structures provide the right class of structures to develop a descriptive complexity theory over $\mathbb{R}$. We substantiate this claim by the following results:

- An appropriate variant of existential second order logic on $\mathbb{R}$-structures, denoted $(\mathrm{SO}\ \exists)_{\mathbb{R}}$, with existential quantification over functions from the finite part of the structure into $\mathbb{R}$, captures $\mathrm{NP}_{\mathbb{R}}$. This is the analogue over $\mathbb{R}$ to Fagin's Theorem.

- We design a fixed-point logic for *partial functions* from finite domains into $\mathbb{R}$

and show that on ordered $\mathbb{R}$-structures, this logic captures $\mathrm{P}_{\mathbb{R}}$. We thus also have a real analogue to the theorem of Immerman and Vardi.

- We investigate appropriate variants of the calculus of primitive recursive and recursive functions over ordered $\mathbb{R}$-structures. The recursive global functions capture precisely computability in polynomial time, whereas the primitive recursive global functions coincide with the functions computable by BSS machines in polynomial time with constant space. As a consequence, there exist primitive recursive global functions which are not in $\mathrm{NC}_{\mathbb{R}}$.
- We exhibit problems that are complete for $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$ under first-order reductions. This will generalize previously known completeness results.

We believe that these results give sufficient evidence that metafinite model theory provides the right framework for a descriptive complexity over $\mathbb{R}$.

2 Computability and complexity theory over the real numbers

Let us briefly describe the computational model over the reals, as introduced by Blum, Shub and Smale [2].

Let $\mathbb{R}^\infty := \bigcup_{k\in\mathbb{N}} \mathbb{R}^k$, or equivalently, the set of functions $X : \mathbb{N} \to \mathbb{R}$ with $X(n) = 0$ for all but finitely many n. For any $X \in \mathbb{R}^\infty$, we call $|X| := \max\{n : X(n) \neq 0\}$ the *length* of X. Note that $\mathbb{R}^\infty \times \mathbb{R}^\infty$ can be identified with $\mathbb{R}^\infty$ in a natural way by concatenation. A *decision problem* is a pair (F, F^+) where $F^+ \subseteq F \subseteq \mathbb{R}^\infty$.

A *Blum-Shub-Smale machine* — in the sequel called a BSS-machine — is essentially a Random Access Machine over $\mathbb{R}$ which can evaluate rational functions at unit cost and whose registers can store arbitrary real numbers.

Definition 2.1 [2] A *BSS-machine* M over $\mathbb{R}$ is given by a finite set I of instructions labelled by $0, \ldots, N$. The input and output spaces are subsets of $\mathbb{R}^\infty$. A configuration is a quadruple $(k, r, w, x) \in I \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^\infty$, where k is the instruction currently executed, r and w are the numbers of the so called "copy registers" (see below) and x describes the content of the registers of the machine. On input $x \in \mathbb{R}^\infty$, the computation is started with configuration $(0, 0, 0, x)$. If $k = N$ the computation stops; in that case the value of x is the output computed by the machine. The instructions are of the following types:

computation: $k : x_0 \leftarrow g_k(x)$ where g_k is a rational function on $\mathbb{R}^\infty$. The next instruction will be $k + 1$, the copy-registers are updated according to either $r \leftarrow r+1$ or $r \leftarrow 0$ and similarly for w. The other registers remain unchanged.

branch: k: **if** $x_0 \geq 0$ **goto** $\beta(k)$ **else goto** $k + 1$. The contents of the registers remain unchanged.

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

Remarks. The rational function g_k evaluated by instruction k depends on a fixed finite collection $x_{i_1}, \ldots, x_{i_s}$ of components of x. The finite set of real constants appearing in the functions g_k are called the *machine constants* of M. Up to an increase of running times by a constant factor, the power of BSS-machines does not change if one restricts the computation-instructions to the basic operations $+, -, \cdot, /$ and assignments $x_0 \leftarrow \alpha$ where α belongs to a fixed finite set of reals (the machine-constants of M).

Definition 2.2 $\mathrm{P}_{\mathbb{R}}$ is the class of all decision problems (F, F^+) for which there exists a BSS-machine over $\mathbb{R}$ deciding, for every given $X \in F$, whether $X \in F^+$; moreover the number of instructions performed during the computation should be bounded by a polynomial in the length of X . The analogue of NP is the class $\mathrm{NP}_{\mathbb{R}}$ of decision problems (F, F^+) for which there exists a decision problem $(G, G^+) \in \mathrm{P}_{\mathbb{R}}$ and a constant k such that $G = \{(X, Y) \in F \times \mathbb{R}^\infty : |Y| \leq |X|^k\}$ and

$$F^+ = \{X \in F : (\exists Y \in \mathbb{R}^\infty)(X, Y) \in G^+\}.$$

Equivalently, $\mathrm{NP}_{\mathbb{R}}$ can be defined as the class of decision problems decided in polynomial time by a *nondeterministic* BSS-machine, i.e. a BSS-machine that can nondeterministically guess tuples $Y \in \mathbb{R}^\infty$ at cost $|Y|$. In particular, during a polynomial-time computation on input X, the concatenation of the tuples guessed by a nondeterministic BSS-machine gives a tuple $Y \in \mathbb{R}^\infty$ with $|Y| \leq |X|^k$ for some fixed $k \in \mathbb{N}$.

Remark. In addition to the above defined "strong" cost measure that assigns unit cost to every operation we will also use the "weak" cost measure of Koiran [21] at some places. In this model, the cost of a single operation also takes into account the degree of the rational function computed so far. While some BSS-machines compute polynomials of exponential degree in a polynomial number of steps, the weak model avoids this effect by only allowing a moderate use of multiplication within polynomial time. By P_w we denote the analogue of $\mathrm{P}_{\mathbb{R}}$ with respect to the weak cost measure.

3 Logic on $\mathbb{R}$-structures

Let us first recall some basic notions from mathematical logic. A *vocabulary* is a set of function and relation symbols each of which comes with some arity $k \in \mathbb{N}$. A *structure* $\mathfrak{A}$ of vocabulary Υ consists of a set A, called the *universe* of $\mathfrak{A}$, and, for each k-ary function symbol $f \in \Upsilon$, a function $f^{\mathfrak{A}} : A^k \to A$, and for each k-ary relation symbol $R \in \Upsilon$, a relation $R^{\mathfrak{A}} \subseteq A^k$. We consider constants as functions of arity 0; to improve readability we usually omit upper indices and write R and f for relations and functions instead of $R^{\mathfrak{A}}$ and $f^{\mathfrak{A}}$.

The structure $\mathfrak{R} = (\mathbb{R}, +, -, \cdot, /, \mathrm{sgn}, <, (c_r)_{r \in \mathbb{R}})$ represents the ordered field of real numbers. We include subtraction and division as primitive operations and assume that every element $r \in \mathbb{R}$ is named by a constant c_r in order to write every rational function $g : \mathbb{R}^k \to \mathbb{R}$ as a term (without quantifiers). In addition we have included the function

$$\mathrm{sgn}(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

which clearly is efficiently computable, but is not a rational function. It turns out, that for logical descriptions it is often more convenient to use the sign function rather than the ordering relation.

We now introduce $\mathbb{R}$-structures which are a special kind of two-sorted structures, i.e. the universe is split into two parts (one of which is the set $\mathbb{R}$ of real numbers). Before giving an exact definition, let us present some examples. They the representation of two decision problems over the reals in terms of $\mathbb{R}$-structures.

Examples. 1) 4-Feasibility. Let F^4 denote the set of all polynomials of degree at most four in n unknowns (where $n \in \mathbb{N}$ is arbitrary) and F^4_{zero} those having a real zero. The problem (F^4, F^4_{zero}) is $\mathrm{NP}_{\mathbb{R}}$-complete [2]. It can be considered as a decision problem on $\mathbb{R}$-structures as follows: Let $A = \{0, \ldots, n\}$. The coefficients of a *homogeneous* polynomial $g \in \mathbb{R}[X_0, \ldots, X_n]$ can now be coded via a function $C : A^4 \to \mathbb{R}$, such that

$$g = \sum_{0 \le i,j,k,\ell \le n} C(i,j,k,\ell) X_i X_j X_k X_\ell .$$

We obtain an arbitrary (not necessarily homogenous) polynomial $f \in \mathbb{R}[X_1, \ldots, X_n]$ of degree four by setting $X_0 = 1$ in g.

Thus, every multivariate polynomial f of degree at most four is represented by the $\mathbb{R}$-structure $(\mathfrak{A}, \mathfrak{R}, \{C\})$ where $\mathfrak{A} = (\{0, \ldots, n\}, <, 0, n)$ and C is a function from A^4 into $\mathbb{R}$.

2) Graphs with real weights. Many important problems arising in practice are defined on graphs with real weights on the vertices and/or edges (e.g. the Travelling Salesman problem, scheduling problems, flow problems, etc.). Weighted graphs can be considered as $\mathbb{R}$-structures, consisting of a graph G together with weight functions assigning real numbers to vertices or edges of G.

The above examples provide typical ways to represent decision problems in terms of $\mathbb{R}$-structures. More explicitely, $\mathbb{R}$-structures consist of a finite structure $\mathfrak{A}$ together with a finite set of functions from $\mathfrak{A}$ into $\mathbb{R}$. They are special kinds of the *metafinite structures* introduced by Grädel and Gurevich. For developing a descriptive complexity theory for computions over the reals it is appropriate to model inputs by $\mathbb{R}$-structures, viewing the finite part as an (ordered) set of indices.

Definition 3.1 Let Υ_a, Υ_w be finite vocabularies where Υ_a may contain relation and function symbols, and Υ_w contains function symbols only. An $\mathbb{R}$-*structure* of vocabulary (Υ_a, Υ_w) is a triple $\mathfrak{D} = (\mathfrak{A}, \mathfrak{R}, W)$ consisting of

(i) a finite structure $\mathfrak{A}$ of vocabulary Υ_a, called the primary part of $\mathfrak{D}$.

(ii) the (infinite) structure $\mathfrak{R} = (\mathbb{R}, +, -, \cdot, /, \mathrm{sgn}, <, (c_r)_{r \in \mathbb{R}})$, called the secondary part of $\mathfrak{D}$.

(iii) a finite set W of functions: $X : A^k \to \mathbb{R}$ interpreting the function symbols in Υ_w. Here A is the universe of $\mathfrak{A}$.

We let $|\mathfrak{D}|$ be the cardinality of A, i.e. of the primary part of $\mathfrak{D}$. A simple class of $\mathbb{R}$-structures is obtained by letting $\mathfrak{A} = (\{1, \ldots, k\}, <)$ be a finite ordered set and W consists of a single unary function $X : \{1, \ldots, k\} \to \mathbb{R}$. Obviously this class models $\mathbb{R}^{\infty}$.

In many cases, decision problems provide a natural ordering on some of their specific input-data (e.g. numbering of variables, of nodes etc.). We will formalize this in the notion of ranked structures.

Definition 3.2 An $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{R}, W)$ is called *ranked* if W contains a function $E : A \to \mathbb{R}$ which is a bijection from A to $\{0, \ldots, |A| - 1\}$. In this case, E is called a *ranking*. It defines a linear order on A. An *m-ranking* is a bijection E_m from A^m to $\{0, \ldots, |A|^m - 1\} \subseteq \mathbb{R}$. Every ranking induces lexicographic m-rankings, for all $m \in \mathbb{N}$.

Since $\mathbb{R}$-structures (and other variants of metafinite structures as well, see [14]) are special kinds of many-sorted structures, they may not seem such a big innovation. Indeed, the real novelty of this approach consists primarily in the logics for such structures, which access the primary and the secondary part in different ways.

We will prove that some of these logics capture exactly the major complexity classes within the BSS approach.

First-order logic. Fix a countable set $V = \{t_0, t_1, \ldots, \}$ of variables. These variables range only over the primary part; we don't use element variables taking values in the secondary part.

The language $\mathrm{FO}_{\mathbb{R}}$ contains, for each vocabulary $\Upsilon = (\Upsilon_a, \Upsilon_w)$, a set $\mathrm{FO}_{\mathbb{R}}(\Upsilon)$ of formulae and terms. Each term takes, when interpreted in some $\mathbb{R}$-structure, values in either the primary part, 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

- The set of index terms is the closure of the set V of variables under applications of function symbols of Υ_a.
- Every constant c_r is a number term.
- If $s_1, \ldots, s_k$ are index terms and X is a k-ary function symbol of Υ_w then $X(s_1, \ldots s_k)$ is a number term.
- If f, g are number terms, then so are $f + g$, $f - g$, $f \cdot g$, f/g and $\mathrm{sgn}(f)$.

Atomic formulae are equalities $s = s'$ of index terms, equalities $f = g$ of number terms, expressions $Ps_1 \cdots s_k$ where P is a k-ary predicate symbol in Υ_a and $s_1, \ldots, s_k$ are index terms, and inequalities $f < g$ of number terms.

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

As we will see in the next section, $\mathrm{FO}_{\mathbb{R}}$ is not a very powerful logic. Therefore we need to enlarge it by additional means. One possibility is that of also allowing quantification over function and relation symbols.

Second-order logic. Second-order logic on $\mathbb{R}$-structures is the logic $\mathrm{SO}_{\mathbb{R}}$ obtained by adding to $\mathrm{FO}_{\mathbb{R}}$ the possibility to quantify over function symbols.

We say that ψ is an *existential second-order sentence* (of vocabulary (Υ_a, Υ_w)) if $\psi = \exists Y_1 \ldots \exists Y_r \varphi$ where φ is a first-order sentence in $\mathrm{FO}_{\mathbb{R}}$ of vocabulary $(\Upsilon_a, \Upsilon_w \cup \{Y_1, \ldots, Y_r\})$. Note once again that we do not allow quantification over functions from $\mathbb{R}$ to $\mathbb{R}$.

4 $\mathrm{NP}_{\mathbb{R}}$ and existential second-order logic

In this section we show that existential second-order logic over $\mathbb{R}$-structures captures the complexity class $\mathrm{NP}_{\mathbb{R}}$. This is an analogue to Fagin's Theorem. To this aim we first have to consider a little closer the way $\mathbb{R}$-structures are represented as inputs for BSS-machines.

Encodings. We showed that $\mathbb{R}^\infty$ can be modelled by $\mathbb{R}$-structures in a natural way. Conversely every $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{R}, W)$ can be encoded by a tuple $e(\mathfrak{D}) \in \mathbb{R}^\infty$ as follows:

Choose a ranking on A and replace all functions and relations in the primary part by the appriorate characteristic functions $\chi : A^k \to \{0, 1\} \subseteq \mathbb{R}$. This gives a structure whose primary part is a plain set A, with functions $X_1, \ldots, X_t$ of the form $X_i : A^k \to \mathbb{R}$ and with the ranking $E : A \to \mathbb{R}$. Each of the functions X_i can be represented by a tuple $x_0, \ldots, x_{m-1} \in \mathbb{R}^m$ where $m = |A|^k$ and $x_i = X(\bar{a}(i))$ where $\bar{a}(i)$ is the i-th tuple in A^k with respect to the k-ranking induced by E. The concatenation of these tuples gives the encoding $e(\mathfrak{D})$. Note that $e(\mathfrak{D})$ depends on the ranking that was chosen.

Obviously, for structures $\mathfrak{D}$ of a fixed finite signature, the length of $e(\mathfrak{D}) \in \mathbb{R}^\infty$ is bounded by some polynomial n^ℓ where $n = |\mathfrak{D}|$ and ℓ depends only on the signature. Thus we can also view $e(\mathfrak{D}) = (x_0, \ldots, x_{n^\ell - 1})$ as a single function $X_{\mathfrak{D}} : A^\ell \to \mathbb{R}$ where $X(\bar{a}(i)) = x_i$ for all $i < n^\ell$. This means that encoding an $\mathbb{R}$-structure in $\mathbb{R}^\infty$ basically means representing the whole structure by a single function (of appropriate arity) from $\{0, \ldots, n-1\}$ into $\mathbb{R}$.

Furthermore this encoding is first-order definable in the following sense:

Lemma 4.1 *For every signature (Υ_a, Υ_w), there is a first-order formula code(X, E) of signature $(\Upsilon_a, \Upsilon_w \cup \{X, E\})$ such that for all $\mathbb{R}$-structures $\mathfrak{D}$ of signature Υ, for all rankings E and for all functions X*

$$(\mathfrak{D}, X, E) \models code(X, E) \quad \text{iff} \quad X = e(\mathfrak{D}).$$

Proof. The formula $code(X, E)$ has to express – for a given $\mathbb{R}$-structure $\mathfrak{D}$, a ranking E and a function $X : A^\ell \to \mathbb{R}$ – that the sequence of all values $X(\bar{v})$, as $\bar{v}$ runs through A^ℓ in the order induced by E, coincides with the concatenation of the tuples $\bar{x}_f \in \mathbb{R}^\infty$ that represent the functions and (the characteristic functions of) the relations of $\mathfrak{D}$. The only difficulty is to determine which function or relation value of $\mathfrak{D}$ is encoded by $X(\bar{v})$ codes. But using the ranking E and the induced rankings on the cartesian products A^k it is not hard (but a bit lengthy) to write down, for every symbol f of Υ_a or Υ_w, a first-order formula $\alpha_f(\bar{u}, \bar{v})$ asserting that $X(\bar{v})$ encodes the value of $f(\bar{u})$. The construction of $code(X, E)$ from these formulae is immediate. ∎

Without loss of generality we view a decision problem as a pair (F, F^+) where F and F^+ are sets of $\mathbb{R}$-structures (of some fixed vocabulary) which are closed under isomorphisms, and where $F^+ \subseteq F$. When we say, that a decision problem (F, F^+) of $\mathbb{R}$-structures is in $\mathrm{P}_{\mathbb{R}}$ or $\mathrm{NP}_{\mathbb{R}}$, we actually mean that $(e(F), e(F^+))$ are in $\mathrm{P}_{\mathbb{R}}$ or $\mathrm{NP}_{\mathbb{R}}$, where $e(F) = \{e(\mathfrak{D}) : \mathfrak{D} \in F\} \subseteq \mathbb{R}^\infty$ and similarly for F^+.

The next lemma shows that $\mathrm{FO}_{\mathbb{R}}$ is a relatively weak logic. Any decision problem expressible in it can be solved in weak polynomial time. Due the fact that P_w is a proper subset of $\mathrm{P}_{\mathbb{R}}$ (see [6]) it follows that first-order logic is strictly weaker than (strong) polynomial time.

Lemma 4.2 *Let (F, F^+) be a decision problem such that $F^+ = \{\mathfrak{D} \in F : \mathfrak{D} \models \psi\}$ for a first-order formula ψ. Then $(F, F^+) \in P_w$.*

PROOF. Assume ψ to be written in prenex normal form, i.e. with all quantifiers in front, and let ρ denote its quantifier free part. Let $\mathfrak{D} \in F$ be an input to be decided. Due to the fact that all quantifiers range over the primary part only and since their number in ψ is fixed (independently of $\mathfrak{D}$), given an input $\mathfrak{D}$ there is only a polynomial number of different assignments for the variables in ψ. Evaluate ρ for all of them. Every term occurring inside ρ has constant degree (independent of $\mathfrak{D}$) and can thus be evaluated in weak polynomial time. Therefore the truth-value of ρ on any of the polynomially many inputs can be computed in weak polynomial time, thus yielding the same to be true for ψ. ∎

When $(F, F^+) \in \mathrm{NP}_{\mathbb{R}}$ we have a decision problem $(G, G^+) \in \mathrm{P}_{\mathbb{R}}$, and we can view the structures in G as expansion of the structures $\mathfrak{D} \in F$ by a k-ary function $Y : A^k \to \mathbb{R}$.

The following theorem is the analogous result in the real setting to Fagin's Theorem.

Theorem 4.3 *Let (F, F^+) be a decision problem of $\mathbb{R}$-structures. Then the following two statements are equivalent.*

(i) $(F, F^+) \in \mathrm{NP}_{\mathbb{R}}$;

(ii) there exists an existential second-order sentence ψ such that $F^+ = \{\mathfrak{D} \in F : \mathfrak{D} \models \psi\}$.

PROOF. It is easy to see that *(ii)* implies *(i)*: Suppose that $\psi = \exists Y_1 \cdots Y_r \varphi$. Given an input structure $\mathfrak{D} \in F$, guess assignments for all functions Y_i and evaluate φ on $(\mathfrak{D}, Y_1, \ldots, Y_r)$ in weak polynomial time, according to Lemma 4.2.

For the converse, let $(F, F^+) \in \mathrm{NP}_{\mathbb{R}}$ and let (G, G^+) be the corresponding problem in $\mathrm{P}_{\mathbb{R}}$, with $G = \{(\mathfrak{D}, Y) : \mathfrak{D} \in F,\ Y \text{ a } k\text{-ary function}\}$ and $F^+ = \{\mathfrak{D} \in F : \exists Y((\mathfrak{D}, Y) \in G^+)\}$.

Let M be a polynomial-time BSS-machine deciding (G, G^+), and m be a natural number such that M stops on $(\mathfrak{D}, Y)$ after less than n^m steps and uses at most $n^m - 3$ registers where $n = |\mathfrak{D}|$.

We first suppose that we have a ranking $E : A^m \to \mathbb{R}$ available. From E, the induced m-ranking E_m is first-order definable: we can identify the element in A of maximal rank and thus have the number term n available; we then can use $E_m(\bar{t})$ as an abbreviation for

$$E(t_1)n^{m-1} + \cdots E(t_{m-1})n + E(t_m).$$

We can then identify A^m with the initial subset $\{0, \ldots, n^m - 1\}$ of $\mathbb{N}$. Thus, in the formulae to be constructed below, m-tuples $\bar{t} = t_1, \ldots, t_m$ of variables are considered to range over natural numbers $t < n^m$. Conditions like $\bar{t} = 0$ or $\bar{t} = \bar{s} + \bar{s}'$ can then be expressed by $\mathrm{FO}_{\mathbb{R}}$-formulae of vocabulary $\{E\}$.

The computation of M on a given input $(\mathfrak{D}, Y)$ can be represented by a function $Z : A^{2m} \to \mathbb{R}$, as follows.

- $Z(0, \bar{t})$ is the instruction executed by M at time $\bar{t}$.
- $Z(1, \bar{t})$ and $Z(2, \bar{t})$ are the indices of the read and the write-registers of M at time $\bar{t}$.
- $Z(\bar{j} + 3, \bar{t})$ is the content of register $\bar{j}$ at time $\bar{t}$.

We construct a first-order formula ψ with the property that for all ranked structures $(\mathfrak{D}, Y) \in G$ and all Z, we have that $(\mathfrak{D}, Y, Z) \models \psi$ iff Z represents an accepting computation of M on $e(\mathfrak{D}, Y)$.

We first have to express that for time $t = 0$, the function Z encodes the input configuration of M on $(\mathfrak{D}, Y)$. Thus we need a subformula, stating that $Z(i, 0) = 0$ for $i = 0, 1, 2$ and that the values $Z(\bar{j}+3, 0)$ encode the input $(\mathfrak{D}, Y)$. By Lemma 4.1 this can be expressed in first-order logic.

Second, we have to ensure that for every $t < n^m - 1$, if the sequence $\langle Z(\bar{j}, \bar{t}) : \bar{j} = 0, \ldots, n^m - 1\rangle$ represents a configuration of M, then the sequence of values $Z(\bar{j}, \bar{t} + 1)$ represents the successor configuration. The formula asserting this has the form

$$\forall \bar{t} \bigwedge_{k=0}^{N} (Z(0, \bar{t}) = k \ \rightarrow \ \varphi_k)$$

where φ_k describes transitions performed by instruction k.

Consider for example a computation instruction $k : x_0 \leftarrow g(x_0, \ldots, x_\ell)$, and assume in addition that it increases the index of the read-register by 1 and sets back the index of the write register to 0.

The formula φ_k then has to express the following

- $Z(0, \bar{t} + 1) = k + 1$ (the next instruction is $k + 1$);
- $Z(1, \bar{t} + 1) = Z(1, \bar{t}) + 1$ (the read-register index is increased by 1);
- $Z(2, \bar{t} + 1) = 0$ (the write-register index is set back to 0);
- $Z(3, \bar{t} + 1) = g(Z(3, \bar{t}), Z(4, \bar{t}), \ldots, Z(\ell + 3, \bar{t}))$ (into register 0, M writes the result of applying the rational function g to the register contents at time $\bar{t}$).
- $Z(\bar{j}, \bar{t} + 1) = Z(\bar{j}, \bar{t})$ for all $\bar{j} > 3$ (the other registers remain unchanged).

Clearly, these conditions are first-order expressible. It should be noted that whenever $f_0, \ldots, f_\ell$ are number terms and $g : \mathbb{R}^\ell \to \mathbb{R}$ is a rational function, then $g(f_0, \ldots, f_\ell)$ is also a number term.

For another example illustrating the explicit use of the embbeding function, consider a copy instruction $k : x_w \leftarrow x_r$. Here the formula has to express (besides update of the instruction number etc. which are done as above), that the content of the register $Z(2,\bar{t})$ at time $\bar{t}+1$ is the same as the content of the register $Z(1,\bar{t})$ at time $\bar{t}$. This is expressed by the formula

$$\forall \bar{j} \forall \bar{j}'([Z(1,\bar{t}) = E_m(\bar{j}) \wedge Z(2,\bar{t}) = E_m(\bar{j}')]$$
$$\rightarrow Z(\bar{j}'+3, \bar{t}+1) = Z(\bar{j}+3,\bar{t})).$$

To express that M accepts its input, we just have to say that $Z(3, n^m - 1) = 1$ (by convention, the result of the computation, if it is a single number, is stored in register 0).

Combining all these subformulae in the appropriate way, we obtain the desired formula ψ. It then follows that for all structures $\mathfrak{D} \in F$

$$\mathfrak{D} \in F^+ \quad \text{iff } \mathfrak{D} \models (\exists Y)(\exists Z)\psi$$

which proves the theorem for the case of ranked structures.

Finally, we do away with the assumption that the input structures are ranked. Indeed, if no ranking is given on structures $\mathfrak{D} \in F$ we can introduce one by existentially quantifying over the function E and adding a conjunct $\alpha(E)$ which asserts that E is one-one and that for all t with $E(t) \neq 0$ there exists an element s such that $E(s) + 1 = E(t)$. It follows that

$$F^+ = \{\mathfrak{D} \in F : \mathfrak{D} \models (\exists E)(\exists Y)(\exists Z)(\alpha \wedge \psi).$$

■

Logical description of 4-Feasibility. An (SO $\exists$)$_{\mathbb{R}}$ sentence for the 4-feasibility problem (F^4, F^4_{zero}) quantifies two functions $X : A \rightarrow \mathbb{R}$ and $Y : A^4 \rightarrow \mathbb{R}$ where $X(1), \ldots, X(n)$ describes the zero and $Y(u)$ is the partial sum of all monomials up to $u \in A^4$ in $f(X_1, \ldots, X_n)$ (according to the lexicographical order on A^4). Thus (F^4, F^4_{zero}) is described by the following sentence ψ:

$$(\exists X)(\exists Y)\Big(Y(\bar{0}) = C(\bar{0}) \wedge Y(\bar{n}) = 0 \wedge$$
$$\forall \bar{u}(\bar{u} \neq \bar{0} \rightarrow Y(\bar{u}) = Y(\bar{u}-1) + C(\bar{u}) \prod_{i=1}^{4} X(u_i))\Big).$$

Indeed, $\mathfrak{D} \models \psi$ if and only if the polynomial f of degree four defined by $\mathfrak{D}$ has a real zero.

5 A fixed point logic for $\mathbb{R}$-structures

Clearly, if we assume $\mathrm{P}_{\mathbb{R}}$ to be different from $\mathrm{NP}_{\mathbb{R}}$ then existential second-order logic is too strong to capture polynomial time. Thus in order to find a logic for $\mathrm{P}_{\mathbb{R}}$ we have to look for a (probably) weaker one than (SO $\exists$)$_{\mathbb{R}}$. On the other hand we have yet seen in Lemma 4.2 that first-order logic is too weak. We thus have to look for a logic in between. The one to be introduced now is an analogue to so-called *fixed-point logic*, which is well known in finite model theory and there captures polynomial time on ordered structures.

For simplicity, we restrict attention now to *functional* $\mathbb{R}$-structures, or $\mathbb{R}$-*algebras*, whose primary part is a plain set A, i.e. $\Upsilon_a = \varnothing$. This is no loss of generality, since we can replace any relation $P \subseteq A^k$ by its characteristic function, considered as a function $\chi_P : A^k \to \mathbb{R}$.

Furthermore we will allow *partially defined* functions from the primary part into $\mathbb{R}$. Formally we define a *partial* $\mathbb{R}$-*algebra* as an $\mathbb{R}^*$-algebra where $\mathbb{R}^*$ is the extension of $\mathbb{R}$ to $\mathbb{R}^* = \mathbb{R} \cup \{\text{undef}\}$. The basic operations on $\mathbb{R}$ are extended to $\mathbb{R}^*$ by

$$\begin{aligned} a + \text{undef} &= a - \text{undef} = \text{undef} \\ a \cdot \text{undef} &= a/\text{undef} = \begin{cases} 0 & \text{if } a = 0 \\ \text{undef} & \text{if } a \neq 0 \end{cases} \\ \operatorname{sgn}(\text{undef}) &= \text{undef} \end{aligned}$$

We define a *functional fixed-point calculus of number terms* by closing the set of first-order number terms, as defined in section 3, under a *maximization rule* and a *fixed point rule*.

Maximization rule: If $F(s, \bar{t})$ is a number term with free variables $s, \bar{t}$, then

$$\max_s F(s, \bar{t})$$

is also a number term with free variables $\bar{t}$, and the obvious semantic. Maximization may be seen as a functional substitute for existential quantifiers.

The notion of a functional fixed point is more complicated. Fix a signature Υ_w and a function symbol Z not contained in this signature. Let $F(Z, \bar{t})$ be a number term of signature $\Upsilon_w \cup \{Z\}$ and free variables $\bar{t} = t_1, \ldots, t_r$ where r is the arity of Z. We write $F^{\mathfrak{D},Z}(\bar{t})$ for the value of $F(Z, \bar{t})$ for a given interpretation $(\mathfrak{D}, Z)$.

For every $\mathbb{R}^*$-algebra $\mathfrak{D}$ of signature Υ_w, the term $F(Z, \bar{t})$ gives rise to an operator $F^{\mathfrak{D}}$ which updates *partially defined* functions Z as follows:

$$(F^{\mathfrak{D}} Z)(\bar{t}) \;:=\; \textbf{if } Z(\bar{t}) = \text{undef } \textbf{then } F^{\mathfrak{D},Z}(\bar{t}) \textbf{ else } Z(\bar{t})$$

This gives an inductive definition of a sequence of partial functions $Z^j : A^r \to \mathbb{R}$.

$$\begin{aligned} Z^0(\bar{t}) &= \text{undef} \quad (\text{for all } \bar{t}) \\ Z^{j+1} &= F^{\mathfrak{D}} Z^j. \end{aligned}$$

Since the operator $F^{\mathfrak{D}}$ updates Z only at points where Z is undefined, this process becomes saturated after polynomially many iterations: $Z^j = Z^{j+1}$ for some $j < |A|^r$. We denote this fixed point by Z^∞ and call it the *fixed point of* $F(Z, \bar{t})$ *on* $\mathfrak{D}$.

Fixed point rule: Given a number term $F(Z, \bar{t})$ of signature $\Upsilon_w \cup \{Z\}$, a tuple $\bar{t} = t_1, \ldots, t_r$ of variables (where r is the arity of Z), and a tuple $\bar{u} = u_1, \ldots, u_r$ of number terms, we build a new number term

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

of signature Υ_w. Its semantic, on a given $\mathfrak{R}^*$-algebra $\mathfrak{D}$, is $Z^\infty(\bar{u})$.

Definition 5.1 *Functional fixed point logic for $\mathbb{R}$-algebras*, denoted $\mathrm{FFP}_{\mathbb{R}}$, is obtained by closing the set of first-order number terms under the maximization rule, under the basic operations $+, -, \cdot, /, \mathrm{sgn}$ of $\mathbb{R}$ and under the fixed point rule.

Proposition 5.2 *Any function in class $\mathrm{FFP}_{\mathbb{R}}$ can be computed in polynomial time.*

The same techniques (e.g. Ehrenfeucht-Fraïssé games) that apply to fixed point logic on finite structures (see [9]) show that $\mathrm{FFP}_{\mathbb{R}}$ does not capture the full power of polynomial-time BSS-machines. For instance, there is no number term of $\mathrm{FFP}_{\mathbb{R}}$ defining the cardinality of the primary part. However, as in finite model theory, fixed-point logic captures polynomial-time on such structures where we have a ranking (or ordering) available so that the fixed point construction can simulate the cycling of an algorithm through all elements.

Let $\varphi(t_1, \ldots, t_r)$ be any formula. We write $\chi[\varphi(t_1, \ldots, t_r)]$ for the characteristic function of φ, with the understanding that $\chi[F = G]$ and $\chi[F < G]$ are undefined, if at least one of the terms F, G is undefined.

Lemma 5.3 *The characteristic function of every $\mathrm{FO}_{\mathbb{R}}$-formula is definable in $\mathrm{FFP}_{\mathbb{R}}$.*

Proof. This is a simple induction. The only point worth mentioning is that the characteristic functions of equalities $F = G$ or inequalities $F < G$ of two number terms can be defined by

$$\begin{aligned} \chi[F = G] &= 1 - [\mathrm{sgn}(F - G)]^2 \\ \chi[F < G] &= ([\mathrm{sgn}(G - F)]^2 + \mathrm{sgn}(G - F))/2. \end{aligned}$$

■

Note that given a ranking E, the induced m-ranking E_m is definable; we will use the abbreviation $\underline{t}$ for $E_m(\bar{t})$. Also we will use $0, 1, \ldots,$ for the tuples—which are of course definable—which are mapped to $0, 1, \ldots,$ by E_m. Thus, strictly speaking, we use a term $F(i)$ (where $i \in \mathbb{N}$) as an abbreviation for $\max_{\bar{t}} \chi[E_m(\bar{t}) = i]F(\bar{t})$.

Theorem 5.4 *On ranked $\mathbb{R}$-structures, the functions in $\mathrm{FFP}_{\mathbb{R}}$ are exactly those computable in polynomial time.*

Proof. Only the inclusion $\mathrm{P}_{\mathbb{R}} \subseteq \mathrm{FFP}_{\mathbb{R}}$ remains to be shown. We represent a computation of a polynomial-time BSS-machine M by a function $Z : A^{2m} \to \mathbb{R}$ in precisely the same way as in the proof of Theorem 4.3, and prove that this function is inductively definable in the form

$$\mathbf{fp}[Z(\bar{j}, \bar{t}) \leftarrow F(Z, \bar{j}, \bar{t})].$$

We use the fact that there is a $\mathrm{FFP}_{\mathbb{R}}$-term defining the characteristic function of every first-order predicate. The term $F(Z, \bar{j}, \bar{t})$ describes the updates for Z at every step of the computation. It has the form

$$\begin{aligned} F(Z, \bar{j}, \bar{t}) :=\ & \chi[\underline{t} = 0] F_{\mathrm{input}}(\bar{j}) + \\ & + \max_{\bar{s}} \chi[\underline{t} = \underline{s} + 1] \sum_{k=0}^{N} \chi[Z(0, \bar{s}) = k] F_k(Z, \bar{j}, \bar{s}) \end{aligned}$$

where F_{input} and F_k are terms describing the encoding of the input and the updates made by instruction k.

The first-order definability of input encodings (Lemma 4.1) yields that F_{input} is definable in $\text{FFP}_{\mathbb{R}}$.

As an example for a term F_k we consider a computation instruction $k : x_0 \leftarrow g(x_0, \ldots, x_\ell)$, that increases the index of the read- and write-registers by 1. The associated term $F_k(Z, \bar{j}, \bar{s})$ is

$$\begin{aligned}\chi[\underline{j} = 0](k+1) + \chi[\underline{j} = 1 \vee \underline{j} = 2](Z(\bar{j}, \bar{s}) + 1) + \\ +\chi[\underline{j} = 3] g(Z(3, s), \ldots, Z(\ell+3, \bar{s})) + \\ +\chi[\underline{j} > 3] Z(\bar{j}, \bar{s})).\end{aligned}$$

Similarly, for a copy instruction $k : x_w \leftarrow x_r$ we have the following term $F_k(Z, \bar{j}, \bar{s})$:

$$\chi[\underline{j} = 0](k+1) + \chi[\underline{j} \neq 0 \wedge \underline{j} \neq Z(2, \bar{s}) + 3] Z(\bar{j}, \bar{s}) +$$

$$+\chi[\underline{j} = Z(2, \bar{s}) + 3] \max_{\bar{j}'} \chi[\underline{j'} = Z(1, \bar{s}) + 3] Z(\bar{j}', \bar{s}).$$

■

6 Primitive recursive and recursive functions on $\mathcal{R}$-structures

As mentioned in the introduction, it was proved by Gurevich that the calculus of primitive recursive functions defines, when interpreted over finite successor structures, precisely the global functions computable in logspace. However, there are important differences between Boolean and real complexity theory with respect to space-resources: it is a result of Michaux [25] that every BSS-decidable problem over $\mathbb{R}^\infty$ can actually be solved in constant space. Thus it is not obvious what type of functions will play the part of logspace computable ones. A closer analysis reveals that there seems to be an enourmous time trade off in applying Michaux' reduction to get constant space computations. It is not known whether the same can be done loosing only a polynomial amount of time. We show that this question is closely related to the relationship between the recursive and the primitive recursive global functions.

We extend the calculi of primitive recursive and recursive functions in a natural way to global functions on $\mathbb{R}$-structures and prove the main result of this section: the primitive recursive global functions are exactly those being computable in polynomial time and constant space whereas the recursive functions are those computable in polynomial time.

Definition 6.1 A *global function* F of vocabulary Υ with arity (ℓ, s) is a function that assigns to each $\mathbb{R}$-structure $\mathfrak{D} = (\mathfrak{A}, \mathfrak{R}, W)$ a function $F^{\mathfrak{D}} : A^\ell \times \mathbb{R}^s \to \mathbb{R}$. For an argument $(\bar{a}; \bar{x})$ of $F^{\mathfrak{D}}$ we call $\bar{a}$ the *index part* and $\bar{x}$ the *number part* of that argument.

Definition 6.2 A global function F is *computable in time* $t(n)$ if there exists a BSS-machine M which, given $\mathfrak{D}$ and $(\bar{a}, \bar{x})$, computes $F^{\mathfrak{D}}(\bar{a}, \bar{x})$ in at most $t(|\mathfrak{D}|)$ steps.

In the following we consider only ranked $\mathbb{R}$-structures.

Definition 6.3 The *initial functions* (of vocabulary Υ) are

(i) the embedding function $E : A \to \mathbb{R}$;

(ii) the functions of vocabulary Υ and the characteristic functions of relations in Υ.

(iii) the constants 0 and *last*, considered as $(0,0)$-ary global functions into $\mathbb{R}$, whose values on $\mathfrak{D}$ are, respectively, 0 and $|A| - 1$.

(iv) for every $c \in \mathbb{R}$, the constant function c.

(v) the sign function and the basic arithmetic operations $+, -, \cdot, /$

(vi) the projections.

The class of *primitive recursive global functions* on $\mathbb{R}$-structures is the closure of the initial functions under composition and (simultaneous) primitive recursion:

If $G_i, \ldots, G_r$ and $H_1, \ldots, H_r$ are primitive recursive, then so are the functions $F_1, \ldots, F_r$ defined by

$$\begin{aligned} F_i(0, \bar{a}; \bar{x}) &= H_i(\bar{a}; \bar{x}) \\ F_i(\bar{t}+1, \bar{a}; \bar{x}) &= G(\bar{t}, \bar{a}; F_1(\bar{t}, a; x), \ldots, F_r(\bar{t}, \bar{a}; \bar{x})) \end{aligned}$$

(where $\bar{t}+1$ is the successor of $\bar{t}$ with respect to the lexicographic ordering on tuples over A). A predicate is primitive recursive iff its characteristic function is.

Lemma 6.4 *Every number term in* $\mathrm{FO}_{\mathbb{R}}$ *and the characteristic function of every first-order property is primitive recursive. Furthermore, if the functions* $G_1, \ldots, G_k$ *and the predicates* $\varphi_1, \ldots, \varphi_k$ *are primitive recursive then so is the function* $F = \sum_{i=1}^{k} \chi[\varphi_i] G_i$.

In order to speak about BSS-machines working in constant space we have to modify the machines (since at least n registers are needed to represent a structure $\mathfrak{D}$ with $|\mathfrak{D}| = n$).

Definition 6.5 A *BSS-machine with separated input and output device*, shortly a SIO-BSS-machine, is a BSS-machine M having besides its working registers two more blocks of registers, for the input and the output. In addition to the usual operations on the working registers M can read values from input registers or write into output registers (with unit cost). A SIO-BSS machine M works in space $t(n)$ if the number of working registers used on inputs of size n does not exceed $t(n)$.

Theorem 6.6 *A global function is computable by a SIO-BSS machine in polynomial time and constant space iff it is primitive recursive.*

PROOF. A simple induction shows that every primitive recursive global function is computable in polynomial-time with constant space by a SIO-BSS machine.

Conversely, assume that F is a global function computed by an SIO-BSS machine M with k working registers in time n^m. Similarly to the proof of Theorem 4.3 we describe the behaviour of M by a collection $\bar{Z} = Z_0, \ldots, Z_{k+2}$ and X, Y of global functions. Here, for $i = 0, \ldots, k-1$, the value $Z_i^{\mathfrak{D}}(\bar{t}, \bar{a}; \bar{x})$ is the content of working register i at time $\bar{t}$ during the computation of M while calculating $F^{\mathfrak{D}}(\bar{a}; \bar{x})$. The remaining three functions Z_k, Z_{k+1} and Z_{k+2} encode the instruction and the indices of the read and write registers, whereas X, Y encodes the content of the input and output registers. Obviously the values $Z_i(\bar{t}+1, \bar{a}; \bar{x})$ depend in a simple way from the values $\bar{Z}(\bar{t}, \bar{a}; \bar{x})$ and from the content of the input registers.

Using Lemma 6.4 and the fact that the encoding of the input is first-order definable, it is not diffcult (although a bit lengthy) to write down a primitive recursive function describing this update. Finally, it is obvious that content of the output registers, and thus the function F too, can be calculated from $\bar{Z}$ by primitive recursion and projection. ■

Since the primitive recursive global functions on finite successor structures are logspace computable, they are in NC^2. Over the reals the situation is different. As shown in [4], $\mathrm{NC}_{\mathbb{R}} \subsetneq \mathrm{P}_{\mathbb{R}}$, where $\mathrm{NC}_{\mathbb{R}}$ consists of all problems decidable by a polynomial number of BSS-machines working in parallel in time $\log^k(n)$ (we refer to [4] for the exact definition).

Theorem 6.7 *There exists a primitive recursive global function which is not computable in polylogarithmic parallel time by a polynomial number of BSS machines.*

PROOF. We want to exploit Cucker's argument [4]. Consider the vocabulary $\Upsilon = \{a, b\}$, where a, b are constant symbols over the reals (i.e., nullary weight functions). For any $\mathbb{R}$-structure $\mathfrak{D}$ over Υ we define a nullary function $c^{\mathfrak{D}} \in \mathbb{R}$ by

$$c^{\mathfrak{D}} := \begin{cases} 1 & \text{if } a^{2^{|\mathfrak{D}|}} = b \\ 0 & \text{otherwise} \end{cases}$$

Obviously all $c^{\mathfrak{D}}$ arise from a global function F in the sense of Definition 6.1. According to [4] for any $k \in \mathbb{N}$ the function F cannot be computed in parallel with a polynomial number of processors in time $O(\log^k n)$ on an input of size n (in fact, F is the characteristic function of the problem $\mathrm{FEAS} = \{(x_1, \ldots, x_n) \in \mathbb{R}^n | x_1^{2^n} = x_2\}$ considered there). However, F is primitive recursive : for any structure $\mathfrak{D}$ let $g : A \to \mathbb{R}$ be defined by $g(0) = a$, $g(t+1) = g(t)^2$. Then g is primitive recursive and $c = \begin{cases} 1 & \text{if } g(n) = b \\ 0 & \text{otherwise} \end{cases}$ implies the same to be true for F. ■

In the literature, there are a number of different calculi of partial recursive functions that give, when they are interpreted over ordered finite structures, precisely the polynomial-time computable global functions. Gurevich presented several such algebras for PTIME in [15], and we refer to [26, 16, 13] for other similar algebras that achieve the same goal.

Many of these algebras can be extended to algebras of global functions on ordered $\mathbb{R}$-structures that characerize polynomial-time computability in the sense of Blum, Shub and Smale.

Following Gurevich we introduce the concept of global recursive functions on ordered $\mathcal{R}$-structures by adapting the classical Herbrand–Gödel–Kleene equation language, cf. [15]. Let Υ be a vocabulary and consider a set E of finitely many equations $t_1 = s_1, \ldots, t_k = s_k$. Here, t_i are terms $F_j(\bar{h}; \bar{\tau})$ where $F_1, \ldots, F_q$ are function symbols not contained in Υ and $\bar{h}$ and $\bar{\tau}$ are tuples of, respectively, index terms and number terms. The s_i are number terms over vocabulary $\Upsilon \cup \{F_1, \ldots, F_q\}$. For instance, one can reformulate the schema for primitive recursion as such a system.

Definition 6.8 Let $\mathfrak{D}$ be an ordered $\mathbb{R}$-structure of vocabulary Υ. A system E as above *defines a function* $F_i : A^\ell \times \mathbb{R}^s \to \mathbb{R}$ *recursively in* $\mathfrak{D}$ iff for all $(\bar{a}; \bar{x}) \in A^\ell \times \mathbb{R}^s$ there exists at most one $w \in \mathbb{R}$ such that $F_i(\bar{a}; \bar{x}) = w$ can be derived from the identities of the form $g(\bar{b}; \bar{y}) = v$ holding in $\mathfrak{D}$, where g is initial primitive recursive, by finite applications of the following steps:

(i) substituting elements from A for individual variables j and closed number terms for real variables x_i.

(ii) if t is a closed term, $u \in \mathbb{R}$ and $t = u$ has already been proved, then occurences of t can be replaced by u.

E *recursively defines* a global function F if it recursively defines $F^{\mathfrak{D}}$ for every $\mathfrak{D}$.

Remark. Note that we don't require the existence of a total function F such that E is satisfied for all values. Thus, checking values of functions defined by E at points $(\bar{a}; \bar{x})$ does not mean to decide the existential first order theory of the reals.

Theorem 6.9 *Let F be a global function of vocabulary Υ. Then F is recursively defined by a system E iff it is computable in polynomial time by a BSS-machine.*

Proof. For any $\mathbb{R}$-structure $\mathfrak{D}$ over Υ let $F^{\mathfrak{D}} : A^\ell \times \mathbb{R}^s \to \mathbb{R}$ be recursively defined by an equation system E. We assume the equations of E to be strictly ordered. For a given input $(a, x) \in A^\ell \times \mathbb{R}^s$ we assign the values x_i to the variables X_i and try to derive the function values $F(\tilde{a}, x)$ for all $\tilde{a} \in A^\ell$. To this aim the equations are considered one after the other (according to the given order). If any function value of the functions appearing in E can be defined because all actual arguments of an equation can be evaluated to a vector of reals, this is done. After treating once all equations, the algorithm starts again passing through the system E - with the yet defined actual function values used to define new ones. Now for all functions F_j of the system there appear at most polynomially many different arguments during this process. This is due to the fact that the number part of the arguments only consists of closed number terms and the index part can vary over A^ℓ only (and thus takes at most n^ℓ different arguments). It follows that after a polynomial number of runs through E all function values of F are known (or it is known that F is not defined at a certain input). Hence F is computable in polynomial time. Note that because of the assumption that at most one $w \in \mathbb{R}$ exists such that $F_j(\bar{a}, \bar{x}) = w$ can be derived, no ambiguities can appear. The same reasoning guarantees the derived function values not to depend on the order of the equations.

The proof of the converse is essentially similar to that of Theorem 6.6. ∎

7 First-order reductions

It is known, that there exist problems which are NP-complete under first-order reductions or even quantifier-free reductions [8, 11]. On ordered structures, the even weaker so-called projection reductions suffice [18, 19].

In this section we want to strengthen some of the completeness results for classes $\mathrm{NP}_{\mathbb{R}}$ and $\mathrm{P}_{\mathbb{R}}$. We introduce the notion of a $\mathrm{FO}_{\mathbb{R}}$-reduction among (classes of) $\mathbb{R}$-structures, and show that both $\mathrm{NP}_{\mathbb{R}}$ and $\mathrm{P}_{\mathbb{R}}$ contain complete problems under these reductions. To do this we find it convenient to generalize the notion of a number term in $\mathrm{FO}_{\mathbb{R}}$ in the following way: Let $T(\mathrm{FO}_{\mathbb{R}})$ be the set of expressions $t_1\chi[\varphi_1] + \cdots + t_k\chi[\varphi_k]$ where t_i are number terms of $\mathrm{FO}_{\mathbb{R}}$ and $\chi[\varphi_i]$ are the characteristic functions of $\mathrm{FO}_{\mathbb{R}}$-formulae φ_i.

The main idea behind $\mathrm{FO}_{\mathbb{R}}$-reductions is to define the functions and relations of an $\mathbb{R}$-structure by evaluating first-order formulae over another, given one.

Definition 7.1 Let Υ, Υ' be vocabularies for $\mathbb{R}$-structures with $\Upsilon = (\Upsilon_a, \Upsilon_w)$. Let $k \in \mathbb{N}$ be a constant. A *first-order interpretation* (of width k) of $\mathbb{R}$-structures over vocabulary Υ in $\mathbb{R}$-structures over vocabulary Υ' is given by a collection Φ of first-order formulae and terms satisfying the subsequent conditions:

(i) for every m-ary relation $R \in \Upsilon_a$, Φ contains a formula φ_R of signature Υ', with free variables t_{ij} where $i = 1, \ldots, m$, $j = 1, \ldots, k$.

(ii) for every m-ary function $f \in \Upsilon_a, \Phi$ contains a formula φ_f of signature Υ' which corresponds to the $m+1$-ary relation given by f (as indicated under *(i)*).

(iii) for every function $X \in \Upsilon_w$ of arity m, there is a number term $F_X \in T(\mathrm{FO}_{\mathbb{R}})$ of signature Υ' in Φ, depending on km variables t_{ij}.

Such intepretations Φ built up the reductions we are looking for. Given a structure $\mathfrak{D}'$ over Υ' a structure $\mathfrak{D} := \Phi\mathfrak{D}'$ over Υ is constructed. To do so one has to define the universe of $\Phi\mathfrak{D}'$. Then the symbols of Υ_a and Υ_w must be interpreted in $\mathfrak{D}$ using Φ.

If $|\mathfrak{D}'| = n$ we take $|\Phi\mathfrak{D}'| := n^k$. Thus any element t in the universe A of $\Phi\mathfrak{D}'$ can be regarded as a tuple $(t_1, \ldots, t_k)$ where $t_i \in A'$, the universe of $\mathfrak{D}'$.

Let $R \in \Upsilon_a$ be an m-ary relation symbol; we then define

$$R(t^{(1)}, \ldots, t^{(m)}) :\Leftrightarrow \mathfrak{D}' \text{ is a model for } \varphi(t_1^{(1)}, \ldots, t_k^{(1)}, \ldots, t_1^{(m)}, \ldots, t_k^{(m)}).$$

The functions in Υ_a are defined similarly. Finally, if $X \in \Upsilon_w$ is an m-ary function symbol, we interpret

$$X(t^{(1)}, \ldots, t^{(m)}) := F_X(t_1^{(1)}, \ldots, t_k^{(m)}) .$$

The following definition is now straightforward.

Definition 7.2 Let K and K' be sets of $\mathbb{R}$-structures of signatures Υ, Υ', respectively. We say that K' is $\mathrm{FO}_{\mathbb{R}}$-*reducible* to K iff there exists a first-order interpretation Φ such that for every $\mathbb{R}$-structure $\mathfrak{D}'$

$$\mathfrak{D}' \in K' \text{ iff } \Phi\mathfrak{D}' \in K.$$

Remark. $\mathrm{FO}_{\mathbb{R}}$-reducibility is the weakest notion of a reduction among those we have mentioned. Any $\mathrm{FO}_{\mathbb{R}}$-reduction is computable in weak polynomial time. This follows in the same manner as Lemma 4.2.

In [2] it is shown that the problem of deciding the existence of a real zero for a degree four polynomial is $\mathrm{NP}_{\mathbb{R}}$-complete. It follows that the problem AC whether a given algebraic circuit evaluates at least one input to 1 is $\mathrm{NP}_{\mathbb{R}}$-complete, too. We will now show the latter problem is $\mathrm{NP}_{\mathbb{R}}$-complete even under $\mathrm{FO}_{\mathbb{R}}$-reductions. This extends a similar result for the discrete setting, which was established by Gács and Lovász [11]. Let us first describe an algebraic circuit as an $\mathbb{R}$-structure.

Definition 7.3 An *algebraic circuit* is an $\mathbb{R}$-structure $\mathrm{AC} = (\mathfrak{A}, \mathfrak{R}, W)$ such that:

(i) the universe A of the primary part contains the gates of the circuit. The functions and relations consist of

- a binary relation F giving the structure of the network ($F(a, b)$ holds iff the result of gate a is an input for gate b).
- unary relations *ADD, SUB, MULT, DIV, SGN, BOOL, SCALAR, NEG, AND, OR* indicating the type of a gate in the circuit. Here *ADD, ..., DIV* stand for the basic arithmetic operations, *SGN* indicates a computation of a sign, *BOOL* stands for a gate introducing a prescribed truth value from $\{0, 1\}$, whereas *SCALAR* represents the introduction of a scalar from $\mathbb{R}$; *NEG, AND, OR* denote the basic Boolean operations.
- a unary relation *INP* , giving the input gates of the circuit as well as a unary relation *OUT* representing the output gate. (We consider circuits with a single output gate only).
- a unary function *CONST1* , which assigns a truth value to the gates satisfying *BOOL*.

(ii) The set W contains only a single function *CONST2*: $A \to \mathbb{R}$, which gives the scalar constants for those gates satisfying the relation *SCALAR* .

We remark that AC is assumed to code an algebraic circuit correctly, i.e. (A, F) is a directed acyclic graph and every $x \in A$ satisfies exactly one of the relations given under *(i)* above. For simplifying the description (especially the proof of Theorem 7.4) we have included some types of gates not intrinsically necessary. All Boolean operations as well as the BOOL gates could be substituted by the remaining operations. However their consideration is convenient in order to relate algebraic circuits with $SO_{\mathbb{R}}$-formulae (see below).

According to Theorem 4.3 the question whether such a circuit returns the value 1 for at least one input is equivalent to the condition that $\mathrm{AC} \models \psi = \exists Y_1 \ldots \exists Y_r \varphi$ for some $\mathrm{FO}_{\mathbb{R}}$-formula φ of vocabulary $(\Upsilon_a, \Upsilon_w \cup \{Y_1, \ldots, Y_r\})$. (It's cumbersome but easy to write down the whole formula. It suffices to take $r = 1$. Then Y_1 describes the value outputted by a gate after application of the corresponding operation and the formula φ states that the results given by Y_1 correctly reflect the computation of the circuit on the according input).

Theorem 7.4 *The algebraic-circuit problem is complete for the class $NP_{\mathbb{R}}$ under* $\mathrm{FO}_{\mathbb{R}}$*-reductions.*

PROOF. Let (G, G^+) be a decision problem in $\mathrm{NP}_{\mathbb{R}}$, where G is a set of $\mathbb{R}$-structures of vocabulary $\Upsilon := (\Upsilon_a, \Upsilon_w)$. The existential second-order sentence corresponding to (G, G^+) according to Theorem 4.3 is denoted by $\psi := \exists Y_1 \ldots \exists Y_r \varphi$. Our task is to define, given any $\mathfrak{D} \in G$, the relations of an algebraic circuit $\mathrm{AC}_{\mathfrak{D}}$ according to Definition 7.3 by $\mathrm{FO}_{\mathbb{R}}-$ formulae such that the structure $\mathfrak{D} \in G$ belongs to G^+ iff the circuit $\mathrm{AC}_{\mathfrak{D}}$ yields the result 1 on some input.

Let $A := \{1, \ldots, n\}$ be the universe of $\mathfrak{D}$. Without loss of generality we assume that φ does not contain any composed terms $X(\ldots, fs, \ldots)$ where $X \in \Upsilon_w \cup \{Y_1, \ldots, Y_r\}$, $f \in \Upsilon_a$ and s is any index term (otherwise we succesively replace atomic formula α containing such a term by a formula $\exists t(fs = t) \wedge \alpha[fs/t]$ where $\alpha[fs/t]$ is the formula obtained from *alpha* by substituting the variable t for the index term fs. Further, we eliminate equalities $F = G$ and inequalities $F < G$ of number terms in φ by replacing them by their arithmetic form as in the proof of Lemma 5.3.

The idea is to interpret φ as an algebraic circuit. This circuit consists of two different parts: first, all terms appearing in atomic formulae are computed (this can be considered as the "really" algebraic part); in the second part the circuit combines the truth values of the atomic formulae to yield the result (the Boolean part of the circuit). For sake of readability we describe the definitions of all relations for the circuit AC informally. However it is an easy exercise to write down the corresponding $\mathrm{FO}_{\mathbb{R}}$ formulae and number terms explicitly. The universe B of the primary part of AC (i.e. the gates of the circuit) is the set of pairs $(\rho, \bar{a})$ where ρ is a *subformula* or a *term* occurring in φ and $\bar{a} = a_1, \ldots, a_k$ is an assignment to the free variables of ρ. Subformulae and terms in φ are called *parts* of φ. Thus,

$$B := \{(\rho, a_1, \ldots, a_k) : \rho \text{ is a part of } \varphi \text{ with } k \text{ free variables and } a_1, \ldots, a_k \in A\}.$$

Note that the elements of B are built up from so-called *minimal parts* by repeated application of one of the gate-operations. Let $t, t_1, \ldots, t_k$ be variable symbols. Minimal part are

(1) relational atoms $Rt_1 \cdots t_j$ for $R \in \Upsilon_a$.

(2) equations of type $ft_1 \cdots t_j = t$ for $f \in \Upsilon_a$.

(3) the real constants in φ

(4) number terms $X(t_1, \ldots, t_j)$ where $X \in \Upsilon_w$ or X is one of the functions symbols quantified in ψ.

Next, $(\rho, a_1, \ldots, a_k) \in B$ stands in relation F with $(\rho', b_1, \ldots, b_s)$ iff the following conditions hold :

- ρ is a part of ρ'.
- There is no part ρ'' of ρ' such that ρ is a part of ρ' (i.e. ρ is a "maximal strict" part of ρ').

- For those variables which are free both in ρ and ρ', the assignments $\bar{a}$ and $\bar{b}$ coincide.

The unary relation $BOOL$ holds for those $(\rho, \bar{a})$ such that ρ is a minimal part of type (1) or (2), the unary relation $SCALAR$ holds for minimal parts of type (3) and of type (4) if the function symbol is not quantified in ψ (i.e. the values are prescribed by the given structure $\mathfrak{D}$). The relation INP holds for those minimal subformulae of type (4) where the function symbol X appears quantified in ψ (they represent the guessed inputs). OUT is true at gate φ (there is no assignment here, since φ has no free variables).

The relations ADD , ..., SGN are given in the obvious way (e.g. $ADD(\rho, \bar{a})$ holds if ρ is of the form $\rho' + \rho''$. Next, the "Boolean" gates are defined according to Boolean operations in φ. For example, $OR(\rho, \bar{a})$ holds if either ρ is the disjunction of two subformulae or $\rho = \exists t \rho'$ In the latter case the gate $(\rho, \bar{a})$ has inputs from all $(\rho', \bar{a}, b)$, where $b \in A$ are the values assigned to the variable t; o bviously the value computed at an OR-gate is the disjunction (i.e. the maximum) of the values computed at its input gates. For gates of type AND and NEG the definitions are similar. In the last step, the functions *CONST1* and *CONST2* have to be described. *CONST1* is given by a $\mathrm{FO}_{\mathbb{R}}(\Upsilon)$- formula assigning to each minimal subformula which satisfies $BOOL$ the truth value of the corresponding relation. *CONST2* does the same for minimal subformulae satisfying $SCALAR$ (here an expression from $T(\mathrm{FO}_{\mathbb{R}}(\Upsilon))$ is used to extract which element of Υ_w corresponds to the given minimal subformula).

Note that the cardinality of the universe B is polynomially bounded in the size n of the given structure $\mathfrak{D}$. This is because the number of subformulae in φ is constant, whereas there are at most polynomially many different assignments to the free variables of any such subformula. ∎

The preceeding theorem extends the results in [2] and [6] showing that $\mathrm{NP}_{\mathbb{R}}$ admits complete problems even under weaker reductions than those being used there.

It is easy to see that first-order reductions can be computed by algebraic circuits of polynomial size and constant depth (if the gates allow an unbounded fan-in; for bounded fan-in one gets polylogarithmic depth). Cucker and Torecillas [7] proved that the problem of deciding whether a given algebraic circuit evaluates a given real input to a given value is $\mathrm{P}_{\mathbb{R}}$-complete under $\mathrm{NC}_{\mathbb{R}}$-reductions. In fact the $\mathrm{NC}_{\mathbb{R}}$-reductions given there can be refined to first-order reductions, establishing the following result.

Theorem 7.5 *The algebraic circuit evaluation problem is* $\mathrm{P}_{\mathbb{R}}$*-complete under* $\mathrm{FO}_{\mathbb{R}}$*-reductions.*

The proof is a straightforward application of the completeness proof in [7].

References

[1] L. Blum, F. Cucker, M. Shub, S. Smale, *Complexity and Real Computation*, Springer-Verlag, forthcoming.

[2] L. Blum, M. Shub, S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bull. Amer. Math. Soc.**21** (1989), 1–46.

[3] P. Bürgisser, M. Clausen, A. Shokrollahi, *Algebraic Complexity Theory*, forthcoming.

[4] F. Cucker, $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$, Journal of Complexity **8** (1992), 230–238.

[5] F. Cucker, D. Grigoriev, *On the Power of real Turing Machines over Binary Inputs*, NeuroCOLT Technical Report NC-TR-94-7, 1994, ESPRIT Working Group 8556.

[6] F. Cucker, M. Shub, S. Smale, *Separation of Complexity Classes in Koiran's weak model*, Theoretical Computer Science **133** (1994), 3–14.

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

[8] E. Dahlhaus, *Reductions to NP-Complete Problems by Interpretations*, in: E. Börger et al. (Eds.), Logic and Machines: Decision Problems and Complexity, Lecture Notes in Computer Science Nr. 171, Springer (1983), 357–366.

[9] H.D. Ebbinghaus, J. Flum, *Finite Model Theory*, Springer (1995)

[10] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, SIAM-AMS Proc. **7** (1974), 43–73.

[11] P. Gács, L. Lovász, *Some Remarks on generalized Spectra*, Zeitschrift für mathematische Logik und Grundlagen der Mathematik **23** (1977), 547–554.

[12] E. Grädel, *Capturing Complexity Classes by Fragments of Second Order Logic*, Theoretical Computer Science **101** (1992), 35–57.

[13] E. Grädel, Y. Gurevich, *Tailoring recursion for complexity*, Journal of Symbolic Logic **60** (1995), 952–969. See also: Proceedings of 21st Colloquium on Automata, Languages and Programming, ICALP 94, Lecture Notes in Computer Science Nr. 820, Springer (1994), 118–129.

[14] E. Grädel, Y. Gurevich, *Metafinite Model Theory*, in: D. Leivant (Ed.), Logic and Computational Complexity, Springer (1995), 313–366.

[15] Y. Gurevich, *Algebras of feasible functions*, Proc. of 24^{th} IEEE Symposium on Foundations of Computer Science (1983), 210–214.

[16] Y. Gurevich, *Logic and the Challenge of Computer Science*, in: E. Börger (Ed.), Trends in Theoretical Computer Science, Computer Science Press (1988), 1–57.

[17] N. Immerman, *Relational queries computable in polynomial time*, Information & Control 68 (1986), 86-104.

[18] N. Immerman, *Languages that Capture Complexity Classes*, SIAM J. Comput. **16** (1987), 760–778.

[19] N. Immerman, *Expressibility as a Complexity Measure, Results and Directions*, Proc. of 2nd IEEE Conference on Structure in Complexity Theory (1987), 194–202.

[20] N. Immerman, *Descriptive and Computational Complexity*, in: J. Hartmanis (Ed.), Computational Complexity Theory, Proc. of AMS Symposia in Appl. Math. **38** (1989), 75–91.

[21] P. Koiran, *A weak version of the Blum-Shub-Smale model*, Proc. 34th IEEE Symposium on Foundations of Computer Science (1993), 486–495.

[22] T. Lickteig, *On semialgebraic decision complexity*, ICSI Tech.Rep. TR-90-052 and Habilitationsschrift, Universität Tübingen (1990).

[23] K. Meer, *On the Complexity of Quadratic Programming in Real Number Models of Computation*, Theoretical Computer Science **133** (1994), 85–94.

[24] K. Meer, C. Michaux, *A Survey on Structural Real Complexity Theory*, preprint 1995.

[25] C. Michaux, *Une remarque à propos des machines sur* $\mathbb{R}$ *introduites par Blum, Shub et Smale*, C.R.Acad.Sc. Paris, t 309, série I (1989), 435-437.

[26] V. Sazonov, *Polynomial computability and recursivity in finite domains*, Elektronische Datenverarbeitung und Kybernetik **16** (1980), 319–323.

[27] V. Strassen, *Algebraic Complexity Theory*, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. A, Elsevier, Amsterdam 1990, pp. 633–672.

[28] M. Vardi, *Complexity of relational query languages*, Proceedings of 14th ACM Symposium on Theory of Computing (1982), 137-146.

[29] J. von zur Gathen, *Algebraic Complexity Theory*, Ann. Reviews of Computer Science **3** (1988), 317-347.