DIMACS Technical Report 94–8
April 1994

Computing over the Reals with Addition and Order: Higher Complexity Classes[1]

by

Felipe Cucker[2]
Universitat Pompeu Fabra
Balmes 132, Barcelona 08008
SPAIN
e-mail: cucker@upf.es

Pascal Koiran[3]
DIMACS, Rutgers University
Piscataway, NJ 08855
USA
e-mail: koiran@dimacs.rutgers.edu

[1]This work was supported by Esprit Project 8556 (NeuroColt)
[2]Partially supported by DGICyT PB 920498, the ESPRIT BRA Program of the EC under contracts no. 7141 and 6546, projects ALCOM II and PROMotion.
[3]Most of this author's work was performed while he was with LIP, Ecole Normale Supérieure de Lyon, France. Support by an INRIA fellowship is also acknowledged.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

This paper deals with issues of structural complexity in a linear version of the Blum-Shub-Smale model of computation over the real numbers. Real versions of PSPACE and of the polynomial time hierarchy are defined, and their properties are investigated. Mainly two types of results are presented:

- Equivalence between quantification over the real numbers and over $\{0, 1\}$;
- Characterizations of recognizable subsets of $\{0, 1\}^*$ in terms of familiar discrete complexity classes.

The complexity of the decision and quantifier elimination problems in the theory of the reals with addition and order is also studied.

In recent years the study of the complexity of computational problems involving real numbers has been an increasing research area. A foundational paper has been [3] where a computational model —the real Turing machine— for dealing with the above problems is developped. Subsequently, several papers dealt with different restrictions of the real Turing machine. Thus in [4] a different kind of nondeterminism —there called digital— is introduced by requiring the guesses to belong to the set $\{0,1\}$. Immediately after, in [6], it is shown that for additive machines (i.e. real Turing machines that do not perform multiplications or divisions) the classes NP resulting from allowing the use of both kinds of nondeterminism whithin polynomial time coincide. In this last paper, the computational power of additive machines restricted to binary inputs is also studied. The main result states that the boolean complexity classes obtained by allowing deterministic and nondeterministic polynomial time on additive machines are P/*poly* and NP/*poly* respectively, and P and NP if the machine is order-free. The study of this "boolean part" of complexity classes over the reals had already been the central subjet of [7].

This paper can be seen as a continuation of [6] since it deals with the computational power of additive machines when restricted to binary inputs and alternation is allowed. In section 3 the polynomial hierarchy for additive machines is introduced in two different versions according to whether the quantifiers act over real or binary variables (generalizing thus the concepts of full and digital nondeterminism). Then, it is shown that these two versions define the same hierarchy of complexity classes from where it is easily deduced that the boolean part of $\Sigma_k^<$ is $\Sigma_k^P/poly$ and the boolean part of $\Sigma_k^=$ is Σ_k^P where Σ_k^P is the k^{th} level of the boolean polynomial hierarchy of Meyer and Stockmeyer. In order to prove these results, the existence of "small" rational points in polyhedra is previously shown in section 2. In section 4, the oracle characterization of the levels of the polynomial hierarchy is given.

On the other hand, as a consequence of the main result of section 3 all the sets in the polynomial hierarchy can be solved in parallel polynomial time. The question arises of whether this is true for the problem of deciding formulæ with an unbounded number of quantifier alternations. As a first step towards an answer to this question in section 5 it is shown that the boolean part of the class of sets decided by additive machines in parallel polynomial time is PSPACE/*poly* (or PSPACE if the machine is order-free). This is used in the next section to prove that the decision of quantified formulæ in the linear theory of the reals is easier when the quantifiers act over binary values. Finally, in section 7 the boolean complexity of quantifier elimination is studied.

Most of the results of sections 2, 3, and 7 can be seen as generalizations of the results of [10]. This paper studied the complexity of the decision problem for formulas in the fist order theory of the reals with addition and order, on the usual Turing machine model. Our results are more general because we allow real inputs and real constants, whereas in [10] only quantified variables can represent real numbers.

1 Basic concepts and notations

In the following we shall deal with the theory of computability and complexity over the reals introduced in [3]. The ground concepts in our paper are taken from there.

As usual, we denote by $\mathbb{R}^\infty$ the direct sum $\bigoplus_1^\infty \mathbb{R}$. We define the *size* $|x|$ of an element $x \in \mathbb{R}^\infty$ as the largest i such that its i^{th} coordinate x_i is different from zero. We denote by Σ the subset $\{0,1\} \subset \mathbb{R}$ and —according to the custom in boolean complexity— by Σ^* the set of all finite sequences of elements in Σ. Any integer number y can be coded in Σ^* in a sequence whose length is bounded by the height of y which is usually defined to be $\log_2(1+|y|)$. There is a natural inclusion of Σ^* in $\mathbb{R}^\infty$ that we shall freely use.

As in [6] we shall deal with additive machines, i.e. real Turing machines that do not multiply or divide. The main properties of these machines and the basic complexity classes defined using them (mainly $\mathrm{P}^<_{\mathrm{add}}$, $\mathrm{P}^=_{\mathrm{add}}$, $\mathrm{NP}^<_{\mathrm{add}}$ and $\mathrm{NP}^=_{\mathrm{add}}$) can be found in [4] or in [6]. We just remind here that we shall use the superscripts $<$ or $=$ for complexity classes according to whether the machines used to define them branch over order tests or only over equality tests. Thus, for instance $\mathrm{EXP}^<_{\mathrm{add}}$ and $\mathrm{EXP}^=_{\mathrm{add}}$ denote the classes of subsets of $\mathbb{R}^\infty$ accepted by additive machines in single exponential time branching over order and equality respectively. Another concept that we remind from [4] is the digital nondeterminism. This is defined by requiring the guesses to belong to $\{0,1\}$ and the word *full* is used to distinguish the nondeterminism as it was introduced in [3] from the digital one. The corresponding complexity classes for polynomial time are denoted by $\mathrm{NP}^<_{\mathrm{add}}$ and $\mathrm{NP}^=_{\mathrm{add}}$ (see [4] or [6] for details).

We now introduce a notation that will be useful in the sequel. It refers to a concept first studied in [7]: the computational power of real Turing machines over binary inputs.

Definition 1 *Given a class $\mathcal{C}$ of subsets of $\mathbb{R}^\infty$, we shall call its Boolean part the class $\mathrm{BP}(\mathcal{C})$ of subsets of Σ^* obtained by considering for any $S \in \mathcal{C}$ its subset of elements belonging to Σ^*. More precisely,*

$$\mathrm{BP}(\mathcal{C}) = \{S \cap \Sigma^*; S \in \mathcal{C}\}.$$

2 More basic facts on polyhedra

The main result of this section is Theorem 2, which establishes the existence of "small" points in polyhedra. The similar but weaker result of [6] is sufficient for dealing with nondeterministic machines, but not with higher levels in the polynomial hierarchy.

The following linear programming result will be useful (see [9], corollary 7.1i).

Lemma 1 (Caratheodory's theorem) *If a vector $b \in \mathbb{R}^n$ is a positive linear combinations of vectors $a_1, \ldots, a_m \in \mathbb{R}^n$ (i.e., if there exist coefficients $x_i \geq 0$ such that $b = \sum_{i=1}^m x_i a_i$), then there exists a linearly independent subset $\{a_{i_1}, \ldots, a_{i_k}\} \subset \{a_1, \ldots, a_m\}$ such that b is a positive linear combination of $a_{i_1}, \ldots, a_{i_k}$.*

We first consider the simpler case of closed polyhedra. The proof method is borrowed from [8].

Theorem 1 *Let P be a non-empty polyhedron of $\mathbb{R}^n$ defined by a system $Ax \leq b$ of N large inequalities, where the entries of A are integers of size L. There exists $I \subset \{1, \ldots, N\}$ and $x \in P$ of the form*

$$x_i = \sum_{j \in I} u_{ij} b_j, \ i = 1, \ldots, n$$

such that $|I| = n$ and the u_{ij}'s are rationals of size $(Ln)^{O(1)}$.

Proof. Since the original system is satisfiable, it can be transformed into a satisfiable system of the form

$$A'x' = b, \ x' \geq 0 \tag{1}$$

by adding a "slack variable" to each inequality, and changing the sign of some of the variables x_i if necessary (i.e., set $x'_i = x_i$ or $x'_i = -x_i$ in a way that makes (1) satisfiable).

In the $N \times (n + N)$ matrix A', the N columns for the N slack variables of x' form an identity matrix. Hence A' has rank N. By hypothesis, (1) has a solution. It follows from Caratheodory's theorem that there exists also a satisfiable system of the form

$$A''x'' = b, \ x'' \geq 0 \tag{2}$$

where A'' is a $N \times N$ matrix made of N linearly independent columns of A'. The variables x'_j corresponding to columns that do not occur in A'' can be set to 0. Since A'' has full rank, (2) has in fact a unique solution which is given by Cramer's rule: $x''_j = \det(A''_j)/\det(A'')$ for $j = 1, \ldots, N$, where A''_j results from A'' by replacing its j^{th} column by b. Since all except perhaps the n first columns of A'' contain exactly one 1 and else only 0's, this matrix can be brought by a permutation σ of its rows into the form

$$B = \begin{pmatrix} C & 0 \\ D & I \end{pmatrix}$$

where C is a square matrix of size n. The entries of C are integers of size at most L, hence the determinant of B is an integer of absolute value smaller than $n!2^{Ln}$.

Let us consider a variable x''_j corresponding to an original variable x_i (therefore $j \leq n$). The matrix A''_j can be brought by the same permutation of rows σ into the form

$$B_j = \begin{pmatrix} C_j & 0 \\ D_j & I \end{pmatrix}$$

where B_j is obtained from B by replacing its j^{th} column vector by $(b_{\sigma(1)}, \ldots, b_{\sigma(N)})$. Expanding $\det(C_j)$ along its j^{th} column yields the desired result, with $I = \{\sigma(1), \ldots, \sigma(n)\}$. $\square$

Theorem 2 *Let P be a non-empty polyhedron of $\mathbb{R}^n$ defined by a system*

$$A_1 x \leq b_1; \ A_2 x < b_2 \tag{3}$$

of N_1 large inequalities and N_2 strict inequalities, where the entries of A_1 and A_2 are integers of size L. There exists $I_1 \subset \{1, \ldots, N_1\}$, $I_2 \subset \{1, \ldots, N_2\}$ and $x \in P$ of the form

$$x_i = \sum_{j \in I_1} u_{ij} b_{1j} + \sum_{j \in I_2} v_{ij} b_{2j} + w_i, \; i = 1, \ldots, n$$

such that $|I_1| + |I_2| = n$, and the coefficients u_{ij},v_{ij},w_i are rationals of size $(Ln)^{O(1)}$.

Proof. Since P is not empty, for some $\epsilon_0 > 0$ and for any $\epsilon \in]0, \epsilon_0[$, the system

$$A_1 x \leq b_1; \; A_2 x \leq b_2 - \epsilon \tag{4}$$

is satisfiable. Conversely, for any $\epsilon > 0$ a solution of (4) is also a solution of (3). In order to obtain a solution of the desired form, we shall thus apply Theorem 1 to (4) for a well-chosen value of ϵ.

According to Theorem 1, for any $\epsilon \in]0, \epsilon_0[$ the new system (4) has a solution $x(\epsilon)$ of the form

$$x_i(\epsilon) = \sum_{j \in I_1} u'_{ij} b_{1j} + \sum_{j \in I_2} v'_{ij} (b_{2j} - \epsilon) \tag{5}$$

where u'_{ij} and v'_{ij} are rationals of size $(Ln)^{O(1)}$, and $|I_1| + |I_2| = n$. Substituting to x in (4) the right-hand side of (5) shows that every $\epsilon \in]0, \epsilon_0[$ is a solution to a system $\mathcal{S}$ of $N_1 + N_2$ inequalities of the form $z_i \epsilon + y_i \geq 0$, where

$$y_i = \sum_{j \in I_1} u''_{ij} b_{1j} + \sum_{j \in I_2} v''_{ij} b_{2j}$$

and the coefficients z_i, u''_{ij}, v''_{ij} are rationals of size $(Ln)^{O(1)}$. There is only a finite number of such coefficients. Hence we can assume without loss of generality that the sets I_1, I_2 and coefficients z_i, u''_{ij}, v''_{ij} are the same for every $\epsilon \in]0, \epsilon_0[$.

For $\mathcal{S}$ to be satisfied by any $\epsilon \in]0, \epsilon_0[$, every y_i has to be non-negative. If z_i is also non-negative, any $\epsilon > 0$ is a solution to the constraint $z_i \epsilon + y_i \geq 0$. If $z_i < 0$, any $\epsilon \leq -y_i/z_i$ is a solution. Let $J = \{i; z_i < 0\}$. If $J = \emptyset$, we set $\epsilon = 1$. Otherwise, we set $\epsilon = \min_{i \in J} -y_i/z_i$. The desired solution to (3) is $x(\epsilon)$. □

3 The polynomial hierarchy

The polynomial hierarchy defined below extends the power of nondeterminism by allowing alternation. This is done for the full version of nondeterminism where any real number can be guessed.

Definition 2 *A real language $L \subset \mathbb{R}^\infty$ is in $\Sigma_k^=$ if there exists a set $A \in \mathrm{P}_{\mathrm{add}}^=$ and polynomial functions $p_1, \ldots, p_k$ such that for any $u \in \mathbb{R}^n$, the following condition holds:*

$$u \in L \Leftrightarrow Q_1 x_1 \in \mathbb{R}^{p_1(n)} \ldots Q_k x_k \in \mathbb{R}^{p_k(n)} (x_1, \ldots, x_k, u) \in A \tag{6}$$

In this equation, the quantifiers $Q_i \in \{\exists, \forall\}$ alternate, starting with $Q_1 = \exists$.

The class $\Pi_k^=$ is defined by the same condition except that the first quantifier is universal ($Q_1 = \forall$). One can also define the classes $\Sigma_k^<$ and $\Pi_k^<$ in the obvious way by assuming only that $A \in \mathrm{P}_{\mathrm{add}}^<$. The polynomial hierarchy $\mathrm{PH}_{\mathrm{add}}^=$ (resp. $\mathrm{PH}_{\mathrm{add}}^<$) is defined as the union of $\Sigma_k^=$ (resp. $\Sigma_k^<$) for all k.

A similar hierarchy can be constructed for digital nondeterminism.

Definition 3 *The classes $\mathrm{D}\Sigma_k^=$, $\mathrm{D}\Sigma_k^<$, $\mathrm{D}\Pi_k^=$ and $\mathrm{D}\Pi_k^<$ are obtained from $\Sigma_k^=$, $\Sigma_k^<$, $\Pi_k^=$ and $\Pi_k^<$ by the following transformation: replace condition (6) by the condition*

$$u \in L \Leftrightarrow Q_1 x_1 \in \{0,1\}^{p_1(n)} \ldots Q_k x_k \in \{0,1\}^{p_k(n)} (x_1, \ldots, x_k, u) \in A$$

Theorem 3 *The following identities hold:*

- $\mathrm{D}\Sigma_k^< = \Sigma_k^<$;
- $\mathrm{D}\Pi_k^< = \Pi_k^<$;
- $\mathrm{D}\Sigma_k^= = \Sigma_k^=$;
- $\mathrm{D}\Pi_k^= = \Pi_k^=$.

Proof. The second indentity follows from the first and the fourth from the third, since $\Pi_k^? = \text{co}-\Sigma_k^?$ and $\mathrm{D}\Pi_k^? = \text{co}-\mathrm{D}\Sigma_k^?$ (here, the "?" symbol stands either for "$<$ " or for "$=$"). We shall first prove by induction on k that $\Sigma_k^< = \mathrm{D}\Sigma_k^<$. Assume that this identity holds for every $i < k$, and let $L \in \Sigma_k^<$. Condition (6) holds for some language $A \in \mathrm{P}_{\mathrm{add}}^<$ and polynomials $p_1, \ldots, p_k$. The $k-1$ internal quantifiers can be eliminated by induction hypothesis: there exists a set $B \in \mathrm{P}_{\mathrm{add}}^<$ and polynomial functions $q_2, \ldots, q_k$ such that for all $u \in \mathbb{R}^n$,

$$\begin{aligned}&u \in L \Leftrightarrow \\ &\exists x_1 \in \mathbb{R}^{p_1(n)} Q_2 x_2 \in \{0,1\}^{q_2(n)} \ldots Q_k x_k \in \{0,1\}^{q_k(n)} (x_1, \ldots, x_k, u) \in B\end{aligned}$$

where the quantifiers $Q_i \in \{\exists, \forall\}$ alternate, starting with $Q_2 = \forall$. Let $\mathcal{B}$ be an additive machine recognizing B in polynomial time. At any time during a computation of $\mathcal{B}$, the content of any coordinate z of its state space is a linear function

$$z = \sum_{i=1}^{p_1(n)} a_i x_{1i} + \sum_{i=1}^{n} b_i u_i + \sum_{i}^{p} c_i \alpha_i + d \tag{7}$$

of its real inputs x_1 and u, and of its real constants $\alpha_1, \ldots, \alpha_p$. The coefficients a_i, b_i,c_i and d are rationals of polynomial size. By enumerating all the accepting computation paths of $\mathcal{B}$, the condition $(x_1, \ldots, x_k, u) \in B$ can be replaced by an (exponential size) boolean formula. The literals of this formula are conditions of the form $z \geq 0$ or $z < 0$, where z is of the form (7). The condition

$$Q_2 x_2 \in \{0,1\}^{q_2(n)} \ldots Q_k x_k \in \{0,1\}^{q_k(n)} (x_1, \ldots, x_k, u) \in B$$

can in turn be considered as a boolean formula, and put in disjunctive normal form (note that this applies also to the case $k = 1$). The conjunctions $S_1, \ldots, S_l$ in this DNF are systems of strict and large inequalities. Hence $u \in L$ if and only if one of the S_i is satisfiable in x_1. These systems may be very large, but their coefficients are of polynomial size. By Theorem 2, if one of the systems is satisfiable, it must have a solution of the form

$$x_{1i} = \sum_{j=1}^{n} b_{ij} u_j + \sum_{j=1}^{p} c_{ij} \alpha_j + d_i \tag{8}$$

where the coefficients b_{ij}, c_{ij} and d_i are rationals of polynomial size. Hence the quantification $\exists x \in \mathbb{R}^{p_1(n)}$ in (6) can be in fact restricted to those x_1 of the form (8). This shows that $\Sigma_k^< \subset \mathrm{D}\Sigma_k^<$, since all the candidate solution for x_1 can be enumerated by guessing a polynomial number of bits. The inclusion $D\Sigma_k^< \subset \Sigma_k^<$ is obvious.

This argument also shows that $\Sigma_k^= \subset \mathrm{D}\Sigma_k^=$. Indeed, if $B \in \mathrm{P}_{\mathrm{add}}^=$, the candidate solutions for x_1 are again of the form (8), and the condition $(x_1, \ldots, x_k, u) \in B$ can be tested in polynomial time by a real order-free Turing machine. □

Corollary 1 *The following equalities hold:*

- $\mathrm{BP}(\Sigma_k^<) = \Sigma_k/poly$;
- $\mathrm{BP}(\Pi_k^<) = \Pi_k/poly$;
- $\mathrm{BP}(\Sigma_k^=) = \Sigma_k$;
- $\mathrm{BP}(\Pi_k^=) = \Pi_k$.

Proof. These identities follow from the equalities $\mathrm{BP}(\mathrm{P}_{\mathrm{add}}^<) = \mathrm{P}/poly$ and $\mathrm{BP}(\mathrm{P}_{\mathrm{add}}^=) = \mathrm{P}$ shown in [6]. Indeed, when $x_1, \ldots, x_k, u$ are all boolean variables, the conditions $(x_1, \ldots, x_k, u) \in A$ for $A \in \mathrm{P}_{\mathrm{add}}^<$ (respectively, $A \in \mathrm{P}_{\mathrm{add}}^=$) or for $A \in \mathrm{P}/poly$ (respectively, $A \in \mathrm{P}$) are equivalent. □

4 On the definition of the polynomial hierarchy

In section 3 we have introduced the polynomial hierarchy —for both kinds of nondeterminism— by means of quantifier alternations. In complexity theory, it is usually done differently. The hierarchy is introduced in a recursive way using oracle machines, and the characterization of its classes in terms of quantifier alternations is then shown as a theorem (see [1] ch.8). In this section we shall prove that this equivalence also holds in our setting. We shall only do it for the ordered case, but the main result in this section also holds in the order–free case.

Let us consider the classes $\widetilde{\Sigma_k^<}$ and $\widetilde{\mathrm{D}\Sigma_k^<}$, for $k \geq 0$ defined in the following way:

i) $\widetilde{\Sigma_0^<} = \widetilde{\mathrm{D}\Sigma_0^<} = \mathrm{P}_{\mathrm{add}}^<$

ii) For any $k > 0$, $\widetilde{\Sigma_k^<} = \mathrm{NP}_{\mathrm{add}}^<(\widetilde{\Sigma_{k-1}^<})$, and $\widetilde{\mathrm{D}\Sigma_k^<} = \mathrm{DNP}_{\mathrm{add}}^<(\widetilde{\mathrm{D}\Sigma_{k-1}^<})$

where by $\mathrm{NP}_{\mathrm{add}}^<(\mathcal{C})$ we mean the class of sets decided by an $\mathrm{NP}_{\mathrm{add}}^<$ machine using as oracle a set in $\mathcal{C}$. Also, for any k, we define $\widetilde{\Pi_k^<}$ to be co–$\widetilde{\Sigma_k^<}$ and $\widetilde{\mathrm{D}\Pi_k^<}$ to be co–$\widetilde{\mathrm{D}\Sigma_k^<}$

The following result shows that the above definition give the same classes for digital and full nondeterminism and that moreover, these classes coincide with the ones already defined in section 3.

Theorem 4 *For every $k \geq 0$ we have*

a) $\widetilde{\Sigma_k^<} = \Sigma_k^<$

b) $\widetilde{\Pi_k^<} = \Pi_k^<$

c) $\widetilde{\mathrm{D}\Sigma_k^<} = \Sigma_k^<$

d) $\widetilde{\mathrm{D}\Pi_k^<} = \Pi_k^<$

Proof. Part *a)* is done in the same way as theorem 8.2. in [1] and part *b)* trivially follows.

Parts *c)* and *d)* will be shown togheter by induction on k. If $k = 0$ the result is certainly true. Let us then consider $k > 0$ and let L be a set in $\Sigma_k^<$. We know then that $L \in \mathrm{D}\Sigma_k^<$ because of theorem 3 and thus, there exists a relation R decidable in $\mathrm{P}_{\mathrm{add}}^<$ and polynomials $p_1, \ldots, p_k$ such that for every $u \in \mathbb{R}^\infty$

$$u \in L \Leftrightarrow Q_1 x_1 \in \{0,1\}^{p_1(n)} \ldots Q_k x_k \in \{0,1\}^{p_k(n)} R(x_1, \ldots, x_k, u)$$

where $Q_1 = \exists$ and the Q_i alternate. Let us define the set B to be the subset of elements $(u, x_1) \in \mathbb{R}^\infty \times \mathbb{R}^\infty$ satisfying

$$Q_2 x_2 \in \{0,1\}^{p_2(n)} \ldots Q_k x_k \in \{0,1\}^{p_k(n)} R(x_1, \ldots, x_k, u) \wedge x_1 \in \Sigma^*$$

Clearly, $B \in \Pi_{k-1}^<$ and by induction hypothesis it belongs then to $\widetilde{\mathrm{D}\Pi_{k-1}^<}$. Now, the additive machine given by

input u
begin
  guess $x_1 \in \{0,1\}^{p_1(n)}$
  ACCEPT iff $(u, x_1) \in B$
end

decides L in digital nondeterministic polynomial time using B as oracle. Thus, $L \in \widetilde{\mathrm{D}\Sigma_k^<}$.

Let us see the reversed inclusion. To do so, we consider a set $L \in \widetilde{\mathrm{D}\Sigma_k^<}$. Such a set can then be decided in digital nondeterministic polynomial time using an oracle B in $\widetilde{\mathrm{D}\Sigma_{k-1}^<}$. Now, by induction hypothesis $B \in \Sigma_{k-1}^<$ and since a digital guess is a particular kind of guess we deduce that $L \in \widetilde{\Sigma_k^<}$.

By part *a)*, we then have that $A \in \Sigma_k^<$. Part *d)* follows by taking complements. □

5 On additive circuits and PSPACE

In this section we define a model for parallel computation and a complexity class resulting from a polynomial time bound on this model. Then, the boolean part of this class is computed and some consequences are given.

Definition 4 *An* additive circuit $\mathcal{C}$ *over* $\mathbb{R}$ *is a directed acyclic graph where each node has indegree 0,1 or 2. Nodes with indegree 0 are either labeled as inputs or with elements of* $\mathbb{R}$ *(we shall call the last ones constant nodes). Nodes with indegree 2 are labeled with "+", or "−". Finally, nodes with indegree 1 are of a unique kind and are called sign nodes. There is a set of* $m \geq 1$ *nodes with outdegree* 0 *called output nodes. In the sequel the nodes of a circuit will be called* gates.

To each gate we inductively associate a function of the input variables in the usual way. In particular, we shall refer to the function associated to the output gates as the function computed by the circuit. Note that sign gates return 1 if their input is greater or equal to 0, and 0 otherwise. However we shall also deal with order–free circuits i.e. those whose sign gates return 1 if their input is different from 0, and 0 otherwise.

Definition 5 *For an additive circuit* $\mathcal{C}$*, the size* $s(\mathcal{C})$ *of* $\mathcal{C}$*, is the number of gates in* $\mathcal{C}$*. The depth* $d(\mathcal{C})$ *of* $\mathcal{C}$*, is the length of the longest path from some input gate to some output gate.*

Definition 6 *Given an additive circuit* $\mathcal{C}$*, the* canonical encoding *of* $\mathcal{C}$ *is a sequence of* 4*-tuples of the form* $(g, op, g_l, g_r) \in \mathbb{R}^4$ *where* g *represents the gate label,* op *is the operation performed by the gate,* g_l *is the gate which provides the left input to* g *and* g_r *its right input. By convention* g_l *and* g_r *are 0 if gate* g *is an input gate, and* g_r *is 0 if gate* g *is a sign gate (whose input is then given by* g_l*) or a constant one (the associated constant being then stored in* g_l*). Also, we shall suppose that the first* n *gates are the input ones and the last* m *the output ones.*

Definition 7 *Let $\{\mathcal{C}_n\}_{n\in\mathbf{N}}$ be a family of additive circuits. We shall say that the family is P–uniform if there exists an additive real Turing machine M that generates the i^{th} gate of the encoding of $\mathcal{C}_n$ with input (i,n) in time polynomial in n.*

Definition 8 *We shall say that a set S can be decided in* additive parallel polynomial time *($S \in \mathrm{PAR}^{<}_{\mathrm{add}}$ for short) when there is a P–uniform family of circuits $\{\mathcal{C}_n\}$ having depth polynomial in n and such that $\mathcal{C}_n$ computes the characteristic function of S restricted to inputs of size n. If the circuits $\mathcal{C}_n$ are order-free as well as the machine that computes them, we shall say that the set S belongs to $\mathrm{PAR}^{=}_{\mathrm{add}}$.*

A first result concerning this new class is the following trivial proposition.

Proposition 1 *The following inclusions of complexity classes hold*

- $\mathrm{PH}^{=}_{\mathrm{add}} \subset \mathrm{PAR}^{=}_{\mathrm{add}} \subset \mathrm{EXP}^{=}_{\mathrm{add}}$
- $\mathrm{PH}^{<}_{\mathrm{add}} \subset \mathrm{PAR}^{<}_{\mathrm{add}} \subset \mathrm{EXP}^{<}_{\mathrm{add}}$

We can now prove the main result of this section.

Theorem 5 *The Boolean part of $\mathrm{PAR}^{<}_{\mathrm{add}}$ is $\mathrm{PSPACE}/poly$.*

Proof. Let $S \in \mathrm{PAR}^{<}_{\mathrm{add}}$ and let us consider its subset $\tilde{S}$ of elements in Σ^*. We shall see that $\tilde{S}$ belongs to $\mathrm{PSPACE}/poly$.

Since S belongs to $\mathrm{PAR}^{<}_{\mathrm{add}}$, there is a family of additive circuits $\{\mathcal{C}_n\}$ having depth bounded by a polynomial $q(n)$. Moreover there is an additive real Turing machine M that given (n,i) produces the i^{th} gate of $\mathcal{C}_n$ within a time that we can suppose to be also bounded by $q(n)$. Let $\alpha_1,\ldots,\alpha_k$ be the constants of M.

With the exception of the constant value for the constant gates, all the remaining values computed by M are positive integers with polynomial size. Without loss of generality we will suppose that M first produces a base two representation of these numbers and then —without using the real constants $\alpha_1,\ldots,\alpha_k$— computes the corresponding integers. This property allows us to suppose that the integer value returned at the end of a computational path only depend on the path itself and not on the constants $\alpha_1,\ldots,\alpha_k$.

On the other hand the constant value $\gamma_{n,i}$ associated to a constant gate can be expressed as a linear combination

$$\gamma_{n,i} = \sum_{j=1}^{k} a_{i,j}\alpha_j$$

of $\alpha_1,\ldots,\alpha_k$ whose coefficients $a_{i,j}$ are integers of polynomial size. Also, let

$$\begin{cases} h_{n,i,j}(\alpha_1,\ldots,\alpha_k) \geq 0 \\ r_{n,i,j}(\alpha_1,\ldots,\alpha_k) < 0 \quad j = 1,\ldots,h_n \end{cases}$$

be the linear equations that determine the computation path followed by M on input (n, i) and let $\Upsilon_{n,i}$ be the linear system of inequations resulting by replacing the α_l by indeterminates X_l for $l = 1, \ldots, k$.

For any n, and for any element $u \in \Sigma^n$, the computation done by $\mathcal{C}_n$ on input u can be described by a set of linear inequations

$$\Xi_{n,u} = \begin{cases} f_{n,u,i}(\gamma_{n,1}, \ldots, \gamma_{n,i_n}) \geq 0 \\ g_{n,u,i}(\gamma_{n,1}, \ldots, \gamma_{n,i_n}) < 0 \end{cases}$$

where the $f_{n,u,i}$ and the $g_{n,u,i}$ are linear functions whose coefficients have polynomial size (because of the polynomial depth of $\mathcal{C}_n$) and the third subindex runs over the sign gates of $\mathcal{C}_n$.

Let us replace in $\Xi_{n,u}$ each occurrence of a $\gamma_{n,i}$ by its correspondent linear combination $r_{n,i}(X_1, \ldots, X_k)$ and let $\zeta_{n,u}$ be the resulting system of inequations. If we now define

$$\Xi_n = \cup_i \Upsilon_{n,i} \cup (\cup_{u \in \Sigma^n} \zeta_{n,u})$$

we obtain a system of inequations that has the real solution $(\alpha_1, \ldots, \alpha_k)$.

From theorem 2 we deduce the existence of a point $r = (r_1, \ldots, r_k) \in \mathbb{Q}^k$ satisfying the system Ξ_n such that each component has size polynomial in n. Therefore if we consider the machine M^r resulting from replacing the constants $\alpha_1, \ldots, \alpha_k$ by $r_1, \ldots, r_k$, the computations done by M^r over binary inputs can be carried out by a Turing machine in polynomial time. The circuit $\mathcal{C}_n^r$ produced by M^r can be readily transformed into a boolean circuit having polynomial depth.

¿From the classical equivalence between parallel polynomial time and PSPACE in the boolean setting, we deduce that $\tilde{S}$ belongs to PSPACE/*poly*.

On the other hand, and using the same equivalence, one trivially shows that any set in PSPACE/*poly* can be accepted by a P–uniform family of circuits in weak parallel time. □

A first consequence of this theorem is the separation of $\mathrm{PAR}^{<}_{\mathrm{add}}$ from $\mathrm{EXP}^{<}_{\mathrm{add}}$.

Corollary 2 *The inclusion* $\mathrm{PAR}^{<}_{\mathrm{add}} \subset \mathrm{EXP}^{<}_{\mathrm{add}}$ *is strict.*

Proof. The boolean part of $\mathrm{EXP}^{<}_{\mathrm{add}}$ is 2^{Σ^*}, the set of all boolean languages. It strictly contains the boolean part of $\mathrm{PAR}^{<}_{\mathrm{add}}$, which is PSPACE/*poly*. □

A variation in the proof of theorem 5 gives us

Theorem 6 *The Boolean part of* $\mathrm{PAR}^{=}_{\mathrm{add}}$ *is* PSPACE.

6 On the complexity of deciding quantified formulæ: the real setting

The purpose of this and the following section is to study the complexity of the elementary theory of the reals with addition and order. This is the set $\mathrm{Th}(\mathbb{R}, +, -, \leq)$ of first order formulæ build from variables and real constants, additions, subtractions and comparisons that are true over $\mathbb{R}$. In fact, any formula in $\mathrm{Th}(\mathbb{R}, +, -, \leq)$ has the form

$$\varphi(u_1, \ldots, u_n) = Q_1 x_1 \in \mathbb{R} \ldots Q_l x_l \in \mathbb{R}\ B(x_1, \ldots, x_l, u_1, \ldots, u_n) \tag{9}$$

where $Q_i \in \{\exists, \forall\}$ and B is a quantifier-free boolean formula. The literals of B are of the form $t \leq t'$, where t and t' are terms constructed by additions and subtractions from the variables x_i,u_i and from real constants. When $n = 0$, i.e. when there are no free variables, it is said that φ is a sentence.

A traditional problem in complexity theory is to determine the complexity of the decision of logical theories (see [5]). That is, for any theory T, the problem of deciding T consists on, given any sentence ψ in the language, determine whether ψ belongs to T, i.e. whether it is true or not over the reals. The traditional approach considers languages with finite alphabet in order to fit the problem into the boolean framework. In this section we shall deal with the theory $\mathrm{Th}(\mathbb{R}, +, -, \leq)$ (as well as with a particular subset of it) and we shall thus consider the additive real Turing machine —and the complexity classes it determines— as the setting to describe the complexity of deciding this theory.

In the next section we shall consider the theory $\mathrm{Th}(0, 1, +, -, \leq)$ of the reals with addition and order but without real constants. The boolean setting will apply then.

In the preceding sections we have seen that the problem of deciding sentences having a bounded number of quantifier alternations can be reduced to the problem of deciding a sentence of the same kind but whose variables now belong to Σ. One of the main results in this sections states that this is no longer true when the number of quantifier alternations is not bounded. In fact, let us denote by TRAO the theory $\mathrm{Th}(\mathbb{R}, +, -, \leq)$ and by DTRAO its subset of sentences all whose variables x satisfy a constraint of the form $x = 0 \vee x = 1$. We will begin by studying the complexity of DTRAO. In order to do so, let us introduce a somehow technical complexity class whose true meaning will become apparent latter.

Definition 9 *We shall denote by $\mathcal{E}$ the class of subsets of $\mathbb{R}^\infty$ that can be decided by an additive machine M satisfying*

- *M works within exponential time*
- *M works within polynomial space*
- *at any moment of the computation of M over an input x, all the coordinates of the state space of M contain a linear combination of $x_1, \ldots, x_n, a_1, \ldots, a_r$ (where $a_1, \ldots, a_r$ are the constants of M) whose coefficients are integers of polynomial height.*

Lemma 2 *The inclusion* $\mathrm{PAR}^{<}_{\mathrm{add}} \subset \mathcal{E}$ *holds.*

Proof. Given a set S in $\mathrm{PAR}^{<}_{\mathrm{add}}$, let M be the additive machine that generates the circuits $\mathcal{C}_n$. The output of the gates of $\mathcal{C}_n$ are linear combination of the required form. Then, the well known algorithm (see [2] 4.2) for simulating the computation of $\mathcal{C}_n$ using M in polynomial space proves the statement. □

Theorem 7 *The set* DTRAO *is* $\mathcal{E}$*-complete.*

Proof. The set DTRAO belongs to $\mathrm{PAR}^{<}_{\mathrm{add}}$ and then to $\mathcal{E}$.

For the hardness we shall use an argument that closely follows the hardness of QBF for PSPACE (see [1] 3.9).

Let $S \in \mathcal{E}$. Then there is a machine M as in definition 9 deciding S working in time $2^{n^{O(1)}}$ and in space n^k. Let $a_1, \ldots, a_r$ be the constants of M.

For any $x \in \mathbb{R}^\infty$ of size n the configuration of M at time t can be described by a vector $(Y, I, J, X_1, \ldots, X_{n^k})$ containing the current node and the contents of the state space. The values Y, I and J belong to $\mathbb{N}$ and the last two are polynomialy bounded. Moreover, also according to definition 9, the values $X_1, \ldots, X_{n^k}$ are integer linear combinations of $x_1, \ldots, x_n$ and $a_1, \ldots, a_r$ whose coefficients have polynomial height. Thus, we can code any configuration α using an element in Σ^* having polynomial size.

Now, given $x \in \mathbb{R}^n$ and configurations α, β we consider the formulæ

$$\mathrm{Next}(\alpha,\beta),\ \mathrm{Equal}(\alpha,\beta),\ \mathrm{Initial}(\alpha,x),\ \text{and}\ \mathrm{Accept}(\alpha)$$

whose respective meanings are "β is the configuration resulting from α after one step of M", "α and β are the same configuration", "α is the initial configuration of M with input x" and "α is an accepting configuration".

Given x these five formulæ can be constructed in polynomial time. We will now construct a new formula $\mathrm{Access\_2}^m(\alpha,\beta)$ describing the fact that the configuration β is reached from α after 2^m steps of M.

If $m = 0$ then, we have

$$\mathrm{Access\_2}^0(\alpha,\beta) = \mathrm{Equal}(\alpha,\beta) \vee \mathrm{Next}(\alpha,\beta)$$

For $m > 0$ we define $\mathrm{Access\_2}^m(\alpha,\beta)$ in the following way

$$\begin{aligned}&\exists\gamma\forall\alpha'\forall\beta'[(\mathrm{Equal}(\alpha',\alpha) \wedge \mathrm{Equal}(\beta',\gamma)) \vee (\mathrm{Equal}(\alpha',\gamma) \wedge \mathrm{Equal}(\beta',\beta)) \Rightarrow \\ &\mathrm{Access\_2}^{m-1}(\alpha',\beta')]\end{aligned}$$

The length of this formula, once recursively expanded, is linear in m and it can be written down in a number of steps polynomial in m.

The desired reduction can be now simply described since, given an $x \in \mathbb{R}^\infty$ it associates to x the formula

$$\mathrm{Accept}(x) = \exists\alpha\exists\beta(\mathrm{Initial}(\alpha,x) \wedge \mathrm{Accept}(\beta) \wedge \mathrm{Access\_2}^{|x|^{O(1)}}(\alpha,\beta))$$

which is constructed in polynomial time in $|x|$. □

Corollary 3 *The classes* $\mathrm{PAR}_{\mathrm{add}}^{<}$ *and* $\mathcal{E}$ *coincide.*

Corollary 4 *The set* DTRAO *is* $\mathrm{PAR}_{\mathrm{add}}^{<}$*-complete.*

Note that in the boolean case, the classes defined by polynomial bounds on deterministic space or on parallel time coincide. Thus, a set in EXP can be efficiently parallelized if and only if an algorithm using polynomial space can be found that fdecides the problem. This is certainly not the case in our setting since we now have that all problems in $\mathrm{EXP}_{\mathrm{add}}^{<}$ can be solved in exponential time and constant work space (see [6] th. 12). Now, a set in $\mathrm{EXP}_{\mathrm{add}}^{<}$ can be efficiently parallelized if and only if there is a sequential algorithm satisfying the conditions of definition 9.

In fact, using the fact that every set in $\mathrm{EXP}_{\mathrm{add}}^{<}$ can be decided in exponential time and constant work space, we can find for any set S in $\mathrm{EXP}_{\mathrm{add}}^{<}$ an additive machine recognizing S in exponential time whose configurations are described with a constant number of real values. The same argument as for theorem 7 now proves the following result.

Theorem 8 *The set* TRAO *is* $\mathrm{EXP}_{\mathrm{add}}^{<}$*–hard.*

Corollary 5 *The set* TRAO *cannot be reduced in polynomial time to* DTRAO. *In particular, it cannot be solved in parallel polynomial time.*

Proof. It follows from the fact —seen in corollary 2— that the classes $\mathrm{PAR}_{\mathrm{add}}^{<}$ and $\mathrm{EXP}_{\mathrm{add}}^{<}$ are different. □

Open question. Is TRAO $\mathrm{EXP}_{\mathrm{add}}^{<}$–complete? Note that this question reduces to determine whether TRAO can be solved in exponential time.

7 On the complexity of quantifier elimination: the boolean setting

In this section we shall consider some problems related to the complexity of quantifier elimination in the theory of the reals with addition and order, but in a language having a finite alphabet. In fact, we shall deal with formulæ of the form

$$\varphi(u_1,\ldots,u_n) = Q_1 x_1 \in \mathbb{R} \ldots Q_l x_l \in \mathbb{R}\ B(x_1,\ldots,x_l,u_1,\ldots,u_n) \tag{10}$$

where $Q_i \in \{\exists,\forall\}$ and B is a quantifier-free formula like (9) but whose terms are constructed by additions and subtractions from the variables x_i,u_i and from integral constants. It is important to remark that this integer constants are written in base two since we shall suppose that the only constant symbols available in our language are 0 and 1. Thus, our computational model will be the Turing machine and all complexity matters in this section refer to the boolean setting.

The quantifier elimination problem consists of finding a quantifier-free formula equivalent to φ. We shall see that this problem can be solved in polynomial time when the number of quantifier alternations in the sequence $Q_1 \ldots Q_l$ is fixed. However, in order to achieve this result, we need to output formulas of a more general form than the quantifier-free formula B defined above. These formulas, which will be called DQC (for Digitally Quantified Circuits) are more general in two ways. First, we need to introduce additional variables to store the results of intermediate computations, like in a straight-line program or a boolean circuit. We also need to to use boolean quantifiers. A DQC is formally defined as follows.

Definition 10 *A DQC is a formula of the form*

$$G(u_1, \ldots, u_n) = Q_1 x_1 \in \{0,1\} \ldots Q_l x_l \in \{0,1\} \; C(x_1, \ldots, x_l, u_1, \ldots, u_n)$$

where $Q_i \in \{\exists, \forall\}$ and C is an additive circuit having only two different constant gates: 0 and 1.

In our opinion, it is fair to consider that an algorithm outputting a DQC solves the quantifier elimination problem, since it is a completely straightforward task to transform a DQC into an ordinary quantifier-free formula (of course, this transformation takes exponential time and yields a formula of exponential size). In fact, we conjecture that both generalizations are necessary if the quantifier elimination problem is to be solved in polynomial time (even when there is only a single block of quantifiers). We show in theorems 10 and 11 that this conjecture holds in the case of order-free formulas.

Theorem 9 *There exists an algorithm $\mathcal{A}$ with the following properties:*

- *$\mathcal{A}$ transforms every formula of the first-order theory of the reals with addition and order under form (10) into an equivalent digitally quantified circuit.*
- *When the number of quantifier alternations in (10) is fixed, $\mathcal{A}$ runs in polynomial time.*

Proof. This results follows from the proof of Theorem 3. The algorithm eliminates blocks of quantifiers sequentially, starting from the innermost block. A block of the form $\exists x_1 \in \mathbb{R}^{p_1(n)}$, can be eliminated using the fact that it suffices to examine candidate solutions of the form (8). The quantification on x_1 is thus replaced by a quantification on the bits of the coefficients b_{ij}, c_{ij} and d_i. In order to eliminate a block of universal quantifiers, we note that $\forall x \; F$ is equivalent to $\neg \, \exists x \, \neg F$. $\square$

The following result demonstrates the necessity of boolean quantifiers. It is similar to the the $\mathrm{P}^{=}_{\mathrm{add}} \neq \mathrm{NP}^{=}_{\mathrm{add}}$ result of [6]. Let $F_n(x_1, \ldots, x_n)$ be the "knapsack formula":

$$\exists \epsilon \in \{0,1\}^n \; \sum_{i=1}^{n} \epsilon_i x_i = 1$$

Note that strictly speaking, $F_n(x_1, \ldots, x_n)$ is not in the first-order language defined at the beginning of this section, but it is straightforward to transform $F_n(x_1, \ldots, x_n)$ into the equivalent formula

$$\exists \epsilon \in \mathbb{R}^n \exists y \in \mathbb{R}^n \ (\bigwedge_{i=1}^{n} (\epsilon_i = 0 \wedge y_i = 0) \vee (\epsilon_i = 1 \wedge y_i = x_i)) \wedge \sum_{i=1}^{n} y_i = 1$$

having length $O(n \log n)$ and belonging to the language.

Theorem 10 *There is no polynomial-time algorithm transforming every $F_n(x_1, \ldots, x_n)$ into an equivalent non-quantified order-free circuit (a circuit is order-free if it does not contain order gates).*

Proof. In fact, there is no polynomial-size order-free circuit C_n equivalent to $F_n(x_1, \ldots, x_n)$. Assume indeed the opposite. All tests performed by C_n are of the form $l(x) \stackrel{?}{=} 0$, where l is a (non-zero) affine function of the input x. Since C_n can perform only a finite number of tests, there exists some input x such that $F_n(x)$ is false and the answers to the tests $l_1(x) \stackrel{?}{=} 0, \ldots, l_m(x) \stackrel{?}{=} 0$ of C_n on this input are all negative. By hypothesis, x is rejected by C_n. Since C_n is of polynomial size, at least one of the 2^n tests of the form $\sum_{i=1}^{n} \epsilon_i x_i \stackrel{?}{=} 1$ has not been performed. Consider such an ϵ and an input $y \in \mathbb{R}^n$ such that $\sum_{i=1}^{n} \epsilon_i y_i = 1, l_1(y) \neq 0, \ldots, l_m(y) \neq 0$ (y exists since the m last constraints remove only a $(n-2)$-dimensional subspace from the hyperplane satisfying the first constraint). The outcomes of all tests are the same for y as for x, hence C_n rejects y. This is a contradiction, since in fact $F_n(y)$ is true. □

The same argument also shows that circuits are necessary.

Theorem 11 *There is no polynomial-time algorithm transforming every $F_n(x_1, \ldots, x_n)$ into an equivalent digitally quantified order-free formula.*

Proof. We again show that there is no polynomial-size order-free digitally quantified formula of the form

$$G_n(x) = Q_1 y_1 \in \{0,1\} \ldots Q_{p(n)} y_{p(n)} \in \{0,1\} H_n(x, y)$$

equivalent to F_n. Let G_n be such a formula; let T be the set of vectors $(a_1, \ldots, a_n) \neq 0$ such that a term of the form

$$t = \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{p(n)} b_i y_i$$

occurs in H_n, and let $\mathcal{T}$ be the corresponding set of terms. Since H_n is of polynomial size, there exists a non-zero vector $(\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n$ which is not in T. Let x be a point such $F_n(x)$ is false and $t(x, y) \neq 0$ for every $y \in \{0,1\}^{p(n)}$ and every term t in $\mathcal{T}$. Let x' be a point such that $\sum_{i=1}^{n} \epsilon_i x'_i = 1$ and again $t(x', y) \neq 0$ for every $y \in \{0,1\}^{p(n)}$ and every term t in $\mathcal{T}$. G_n cannot be equivalent to F_n since $G_n(x) = G_n(x')$, $F_n(x')$ is true and $F_n(x)$ is false. □

References

[1] J.L. Balcázar, J. Diáz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.

[2] J.L. Balcázar, J. Diáz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1990.

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

[4] F Cucker and M. Matamala. On digital nondeterminism. preprint, 1993.

[5] J. Ferrante and C.W. Rachof. *The Computational Complexity of Logical Theories.* Lect. Notes in Math., vol. 718. Springer Verlag, 1979.

[6] P. Koiran. Computing over the reals with addition and order. Presented at the Barcelona workshop on continuous algorithms and complexity, 1993. To appear in TCS.

[7] P. Koiran. A weak version of the Blum, Shub & Smale model. In *Proc.* 34^{th} *FOCS Symp.*, pages 486–495, 1993.

[8] W. Maass. Bounds for the computational power and learning complexity of analog neural nets. In *Proc. 25th STOC*, pages 335–344, 1993.

[9] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New-York, 1986.

[10] E. D. Sontag. Real addition and the polynomial hierarchy. *Information Processing Letters*, 20:115–120, 1985.