**DIMACS Technical Report 94–10**
**April 1994**

# A Weak Version of the Blum, Shub & Smale model

by

Pascal Koiran[1]
DIMACS
Rutgers University
e-mail: koiran@dimacs.rutgers.edu

[1]Most of this work was performed while the author was with LIP, Ecole Normale Supérieure de Lyon, France. Support by an INRIA fellowship and Esprit Project 8556 (NeuroColt) 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

We propose a weak version of the Blum-Shub-Smale model of computation over the real numbers. In this weak model only a "moderate" usage of multiplications and divisions is allowed. The class of boolean languages recognizable in polynomial time is shown to be the complexity class P/poly. The main tool is a result on the existence of small rational points in semi-algebraic sets which is of independent interest. As an application, we generalize recent results of Siegelmann & Sontag on recurrent neural networks, and of Maass on feedforward nets. A preliminary version of this paper was presented at the 1993 IEEE Symposium on Foundations of Computer Science. Additional results include:

- an efficient simulation of order-free real Turing machines by probabilistic Turing machines in the full Blum-Shub-Smale model;
- an efficient simulation of arithmetic circuits over the integers by boolean circuits;
- the strict inclusion of non-deterministic full polynomial time in weak exponential time.

# 1 Introduction

The real number model of computation is a quite natural and convenient framework for studying numerical computation. It is nonetheless often criticized for being overly unrealistic. The assumption that arbitrary real numbers can be compared, added or multiplied in constant time and infinite precision is the usual target of these criticisms (see [28] for a discussion of this issue). This paper proposes a model of computation (the "weak BSS model") which can be seen as a middle ground between the usual Turing machine model and the real Turing machine of Blum, Shub and Smale [4].

The weak BSS model is obtained from the standard one by dropping the constant cost hypothesis for arithmetic operations. This yields a notion of feasible analog computation which is perhaps more realistic, since the standard notion of discrete polynomial-time computability is recovered when the real parameters defining a real Turing machine happen to be rational. In other words, an algorithm that runs in polynomial time in the weak model is "ideal" in the sense of Renegar, since "it seems that the ideal algorithm is one that provides a "nice" bound in the real number model and which when restricted to sentences with polynomials whose coefficients are rational number also provides a "nice" bound in the bit model" [22].

One important theme of this paper is that when both input and outputs of real machines are discrete, complexity classes in the real number model can be identified to discrete complexity classes. For instance, our main theorem states that the class of boolean languages recognizable in polynomial time in the weak BSS model is P/poly (P/poly is the class of sets recognized in polynomial time by Turing machines using advice functions of polynomial length or, equivalently, by non-uniform boolean circuits of polynomial size [1]). The main tool for this proof is a new result on semi-algebraic sets of independent interest, stating that a semi-algebraic set of non-empty interior defined by "small" equations must contain a "small" rational point. It follows from this characterization that P$\neq$NP in the weak model under a standard complexity-theoretic method. After a first version of this paper was written [17], it was shown in [12] by quite different methods that P$\neq$NP in the weak model without any assumption. The results on boolean inputs and semi-algebraic sets remain nonetheless interesting.

The rest of this paper is organized as follows. After recalling some standard background on semi-algebraic sets in section 2, we present the full and weak BSS models in section 3. The main theorem is stated here, along with simpler results of the same type. The existence of small points in semi-algebraic sets is established in section 4; as explained in section 5, the main theorem follows from this result. Models of computation without order are studied in section 6. It is shown that real Turing machines can be efficiently simulated by ordinary Turing machines in the weak model, and by probabilistic Turing machines in the full BSS model; and that non-uniform arithmetic circuits over the integers can be efficiently simulated by boolean circuits. In section 7, we show that non-deterministic full polynomial time is strictly included in weak exponential time. The strict inclusion was left as an open problem in [12]. We give applications of our results to recurrent and feedforward neural networks in section 8. It is shown that that the class of functions computed by recurrent nets using (possibly discontinuous) piecewise-linear output functions is exactly P/poly. This was previously known only for continuous output functions (see [26] or the extended abstract [27]). For families of feedforward nets we define a notion of uniformity which parallels the standard definition for boolean circuits. We then show that some uniform families of real circuits can be efficiently simulated by families of threshold circuits. This problem was studied recently by Maass [20] in the more general case of non-uniform families of real circuits, but his results apply only to constant depth circuits. This restriction can be removed for uniform circuit families. Suggestions for further work can be found in section 9 and at the end of section 6.

# 2 Preliminaries

In this section, we list the notations used throughout the paper, and recall some standard background on semi-algebraic sets (a comprehensive treatment of this topic can be found in [3] and [5]).

## 2.1 Miscellaneous notations

The set $\mathbb{N}$ of non-negative integers is identified with $\{0,1\}^+$, the set of non-empty words on the alphabet $\{0,1\}$.

The size $s(x)$ of an integer $x \in \mathbb{Z}$ is the number of digits of its radix-2 representation. The size of a rational number $x \in \mathbb{Q}$ is $s(x) = \min\{s(p)+s(q), x = p/q,\ p,q \in \mathbb{Z}\}$. The size of a rational point $x = (x_1,\ldots,x_p) \in \mathbb{Q}^p$ is $s(x) = \sum_{i=1}^p s(x_i)$.

Given a ring $A$, $A[X_1,\ldots,X_p]$ is the ring of polynomials in $p$ variables with coefficients in $A$. When there is no risk of confusion, we write also $A[X]$ instead of $A[X_1,\ldots,X_p]$. The degree of $P \in A[X_1,\ldots,X_p]$ is $d(P)$, and the degree of a rational fraction $R = P/Q$ is $d(R) = d(P) + d(Q)$.

## 2.2 Semi-algebraic sets

**Definition 1** *A basic semi-algebraic set $\mathcal{S} \subset \mathbb{R}^p$ is defined by a conjunction of polynomials equalities and inequalities:*

$$\mathcal{S} = \{x \in \mathbb{R}^p;\ Q_1 > 0 \wedge \ldots \wedge Q_m > 0 \wedge R_1 = 0 \wedge \ldots \wedge R_n = 0\}$$

*where $Q_i, R_i \in \mathbb{Z}[X_1,\ldots,X_p]$ are non-constant polynomials ($m$ and/or $n$ may be equal to zero). A semi-algebraic set is a finite union of basic semi-algebraic set.*

Usually all sign conditions ($P = 0$, $P \leq 0$, $P < 0$, $P \geq 0$ and $P > 0$) are allowed in the definition of a semi-algebraic set, but these two definitions are clearly equivalent.

It is possible to take the $P_i$'s and $R_i$'s in $\mathbb{R}[X]$ instead of $\mathbb{Z}[X]$. The resulting sets will also be called semi-algebraic. It should be clear from the context whether the polynomials have integer or real coefficients.

**Theorem 1 (The Tarski-Seidenberg principle)**
*Let $S' = \{x \in \mathbb{R}^n;\ \exists y \in \mathbb{R},\ (x, y) \in S\}$ be the projection on $\mathbb{R}^n$ of a set $S \subset \mathbb{R}^{n+1}$. If $S$ a semi-algebraic set with coefficients in $\mathbb{Z}$ (respectively, $\mathbb{R}$), $S'$ is a semi-algebraic set with coefficients in $\mathbb{Z}$ (respectively, $\mathbb{R}$).*

An important consequence of the Tarski-Seidenberg principle is the elimination of quantifiers: a set defined by a logical formula in the fist-order theory of the reals is semi-algebraic. These formulas are defined by induction as follows.

**Definition 2 (First-order formulas)**
*$P$ op $0$ is a first-order formula, for $P \in \mathbb{Z}[X_1, \ldots, X_n]$ and op $\in \{<, \leq, =, \geq, >\}$.*
*If $F_1$ and $F_2$ are first-order formulas, $F_1 \vee F_2$, $F_1 \wedge F_2$, $\forall x \in \mathbb{R}\ F_1$ and $\exists x \in \mathbb{R}\ F_1$ are first-order formulas.*

If there are $n$ free variables $x_1, \ldots, x_n$ in a formula $F$, we write also $F(x_1, \ldots, x_n)$ instead of $F$. The subset of $\mathbb{R}^n$ defined by $F$ is

$$S = \{(x_1, \ldots, x_n) \in \mathbb{R}^n; F(x_1, \ldots, x_n) \text{ is true}\}.$$

**Theorem 2 (Elimination of quantifiers)** *The set $S \subset \mathbb{R}^n$ defined by a first-order formula $F(x_1, \ldots, x_n)$ is semi-algebraic.*

*Proof Sketch.* Semi-algebraic sets are clearly closed under boolean operations. The Tarski-Seidenberg principle amounts to the elimination of one existential quantifier. A universal quantifier can be replaced by an existential quantifier since $\neg\exists x\ \neg F$ is equivalent to $\forall x\ F$. $\square$

# 3 Simulating real Turing machines

## 3.1 Presentation of the model

Blum, Shub & Smale proposed a model of computation (the real Turing machine) that can be seen as a RAM machine in which a register can store an arbitrary real number with infinite precision. The constants occurring in the program executed by a real Turing machine (RTM in the sequel) are also arbitrary real numbers, and the basic operations $(+, -, \times, \div, \leq)$ can be performed in constant time.

Throughout the paper, $\alpha_1, \ldots, \alpha_p$ are the non-zero constants occurring in a program.

A computation tree can be associated to a program as follows: each node of the tree is labeled by an instruction. A test node has two children which represent the two possible outcomes of the test. The other internal nodes have a single child. At each node $\mathcal{N}$, the content of a variable $x$ is a rational function $Q_{\mathcal{N},x}(u, \alpha)$ of the input $u \in \mathbb{R}^n$ and the constants $\alpha = (\alpha_1, \ldots, \alpha_p)$.

We shall in fact work in a modified BSS model in which the cost of an arithmetic operation is *not* constant.

**Definition 3** *If $x = Q(u, \alpha)$ after an arithmetic operation $x := y$ op $z$, the cost of this operation in the weak BSS model is the maximum of $d(Q)$ and the maximum size of the coefficients of $Q$.*

This model is weaker than the standard one since the class of polynomial-time computable functions is (presumably) smaller than in the full BSS model. For example, it is possible to output integers of exponential length in polynomial time with a program using only integer constants in the full model. This is clearly impossible in the weak model (even if real constants are allowed, as could be shown by generalizing Theorem 4 to integer-valued outputs).

The weak BSS model is interesting not only because of theorems 4 and 16. It is perhaps a more realistic model of feasible analog computation than the full BSS model since it is not difficult to see that the standard notion of discrete polynomial-time computability is recovered when the constants $\alpha_1, \ldots, \alpha_p$ are rational (see Theorem 3). Note that the perhaps undesirable possibility to create integers of exponential length in polynomial time in the full model is a crucial tool in the recent proof that $\mathrm{P}_R \neq \mathrm{NC}_R$ [8]. This argument fails to separate weak polynomial time from $\mathrm{NC}_R$.

The following easy characterization will be useful.

**Lemma 1** *A function is polynomial-time computable in the weak BSS model if and only if it is polynomial-time computable in the standard BSS model and the rational fractions $Q_{\mathcal{N},x}$ have polynomial degree and coefficients of polynomial size.*

*Proof.* If $F$ is computable in time $T(n)$ in the weak model, it is also computable in time $T(n)$ in the full model and the degrees and coefficient sizes of the rational fractions $Q_{\mathcal{N},x}$ are bounded by $T(n)$.

If $F$ is computable in time $T(n)$ in the full model with degrees and sizes bounded by $S(n)$, $F$ is computable in the weak model in time $T(n)S(n)$. □

The weak BSS model includes as a special case a model in which divisions are forbidden and a variable can be multiplied only by a constant (and not another variable). This simple subcase is sufficient for dealing with the neural nets of section 8.

**Lemma 2** *Let $F$ be a polynomial-time computable function in the standard BSS model without divisions by a program in which a variable is not multiplied by another variable. Then $F$ is polynomial-time computable in the weak BSS model.*

*Proof.* The rational fractions $Q_{\mathcal{N},x}$ are in fact polynomials since no division is performed. Let us show by induction that at time $t$, the content of a variable $x$ is $P(\alpha, u)$, with $d(P) \leq t+1$ and the coefficients of $P$ bounded by $2^t$.

The result clearly holds for $t = 0$. Assume that the result holds at time $t$. At time $t+1$, it suffices to consider the case of a multiplication by a constant $x := \alpha_i y$ or an addition $x := y + z$. Let $y = P(\alpha, u)$ and $z = Q(\alpha, u)$. $d(\alpha_i P) = d(P) + 1$, and the coefficients do not change. $d(P + Q) \leq \max(d(P), d(Q))$, and the coefficients can be at most doubled. □

Let us now introduce some notations. $\mathrm{P}_R$ and $\mathrm{P}_W$ are the classes of real languages, i.e., subsets of

$$\mathbb{R}^\infty = \bigcup_{n=0}^{\infty} \mathbb{R}^n$$

that can be recognized in polynomial time in the BSS model and the weak BSS model, respectively. $\mathrm{NP}_R$ and $\mathrm{NP}_W$ are the non-deterministic counterparts of $\mathrm{P}_R$ and $\mathrm{P}_W$. More precisely, a language $L$ is in $\mathrm{NP}_?$ if there is a language $A \in \mathrm{P}_?$ and a polynomial function $p$ such that for every $n$ and every $u \in \mathbb{R}^n$,

$$u \in L \Leftrightarrow \exists x \in \mathbb{R}^{<p(n)} (x, u) \in A$$

where the symbol "?" stands for $W$ or $R$, and

$$\mathbb{R}^{<p(n)} = \bigcup_{k=0}^{p(n)} \mathbb{R}^k.$$

In fact, one can assume without loss of generality that the machine makes the same number of guesses for all inputs, i.e., that $x$ is in $\mathbb{R}^{p(n)}$ instead of $\mathbb{R}^{<p(n)}$.

## 3.2 Boolean inputs

In the rest of this paper, we are mainly interested in computing boolean functions on $\{0,1\}^*$, or, equivalently, recognizing boolean languages (subsets of $\{0,1\}^*$). We thus address the following question: what can be gained from the availability of arbitrary real constants ? (see section 7 for a result of a different nature.) The boolean part $\mathrm{BP}(\mathcal{C})$ of a class $\mathcal{C}$ of real languages is the restriction of $\mathcal{C}$ to boolean inputs. More precisely,

$$\mathrm{BP}(C) = \{L \cap \{0,1\}^*; L \in \mathcal{C}\}.$$

For instance, $\mathrm{BP}(\mathrm{P}_W)$ is the class of boolean languages which can be recognized in polynomial-time in the weak BSS model. Note that in the $\mathrm{BP}(\mathrm{NP}_W)$ or $\mathrm{BP}(\mathrm{NP}_R)$ cases, the inputs are boolean but the guesses are real numbers.

We will sometimes consider machines which cannot use arbitrary real constants (see section 7). If $\mathcal{C}$ is a class of languages recognized by resource-bounded real machines, $\mathcal{C}(E)$ denotes the class of languages recognized within the same resource constraints by real machines using only constants from the set $E \subset \mathbb{R}$. For example, $\mathrm{P}_R(\mathbb{R}) = \mathrm{P}_R$ and $\mathrm{BP}(\mathrm{P}_W(0,1))$ is the set of boolean languages recognized in weak polynomial time by machines using only the constants 0 and 1.

**Theorem 3** $\mathrm{BP}(\mathrm{P}_W(\mathbb{Q})) = \mathrm{P}$.

*Proof.* If $L \in \mathrm{BP}(\mathrm{P}_W(\mathbb{Q}))$, it follows from Lemma 1 that $L \in \mathrm{P}$ since during a computation, each register of the real Turing machine contains a rational number of polynomial size.

Clearly, $\mathrm{P} \subset \mathrm{BP}(\mathrm{P}_W(\mathbb{Q}))$ since real machines are at least as powerful as Turing machines (in fact, $\mathrm{P} \subset \mathrm{BP}(\mathrm{P}_W(0,1))$). $\square$

We are now in position to state our main result.

**Theorem 4** BP($P_W$) = P/poly.

*Proof.* In order to show that P/poly $\subset$ BP($P_W$), we encode an advice function in the digits $a_i$ of a real constant $\alpha = \overline{0.a_1 a_2 \ldots a_i \ldots}$. Let us choose an encoding scheme such that infinitely many $a_i$'s are equal to 0. The value of $a_1$ can be easily computed since $a_1 = 0 \Leftrightarrow \alpha < 1/2$. In order to compute $a_2$, we shift $\alpha$ to the left with the instruction $\alpha := 2\alpha - a_1$. Now $a_2$ can be computed like $a_1$. By iterating this process, it is possible to retrieve as many digits of $c$ as desired. Since the advice are of polynomial size, the whole decoding process takes polynomial time. According to lemma 2, it actually takes weak polynomial time.

We prove in section 5 that BP($P_W$) $\subset$ P/poly, with the help of a technical result established in section 4. $\square$

**Corollary 1** *If* $P_W = NP_W$, *the polynomial hierarchy collapses at the second level.*

*Proof.* According to Theorem 4, $P_W = NP_W \Longrightarrow NP \subset P/poly$ since $NP \subset BP(NP_W)$.

If NP $\subset$ P/poly, the polynomial hierarchy collapses at the second level [2]. $\square$

As was mentioned in the introduction, it was recently shown in [12] that $P_W \neq NP_W$ holds without any assumption. The decoding argument of Theorem 4 also shows that all boolean languages can be recognized in weak exponential time.

**Theorem 5** BP($EXP_W$) = $2^{\{0,1\}^*}$.

*Proof Sketch.* The truth table of any boolean language can be encoded in the digits of a real constant, and it can be decoded in weak exponential time as in Theorem 4 (there are $2^n$ digits for inputs of size $n$). $\square$

## 3.3 Hardness and feasibility results

In this paper the emphasis is on the solution of discrete problems with real "black boxes". This was surely not the point of view of Blum, Shub & Smale when they proposed their model. They were much more interested in the solution of continuous problems. However our point of view is not too restrictive for hardness results, since an algorithm solving a continuous problem must in particular provide the correct answer for discrete inputs. We can thus give evidence of hardness for the real-valued generalizations of the NP-complete problems of combinatorial optimization, such as the real traveling salesman problem of [4] or the real knapsack problem. More precisely, if one of these problems can be solved in polynomial time in the weak BSS model, NP $\subset$ P/poly according to Theorem 4, and the polynomial hierarchy collapses at the second level. Let us note in passing that no completeness result is known for the real TSP or other similar problems, either in the standard BSS model or in a weaker model. Moreover, some techniques designed for dealing with boolean inputs and real constants can be adapted to the case of real inputs and boolean constants (see section 7).

In our opinion, many (if not all) natural problems that can be solved in polynomial time in the standard BSS model can in fact be solved in polynomial time in the weak model. This is obviously true for very easy problems such as multiplication of polynomials or matrices. Let us give two other non-trivial examples.

The first problem consists of determining the sign of a real polynomial $P$ at the roots of another real polynomial $Q$. Two algorithms for this problem are described in [9] and [22], and it is shown that the conditions of Lemma 1 are satisfied. Hence they run in polynomial time in the weak BSS model. Note that the algorithm of [9] is used in the same reference as a building block for computing the topology of a real algebraic plane curve.

Our second example is simply the resolution of systems of linear equations. This problem can obviously be solved in polynomial time in the BSS model by Gaussian elimination. For rational data, it is known that this procedure also runs in polynomial time in the bit model ([25], theorem 3.3). It can be proven as an intermediate result that during the elimination procedure the coefficients of the system have a very special form: they are quotients of determinants extracted from the matrix of the original system. Note that as a polynomial function of the entries, the determinant of a matrix has polynomial (in fact sub-linear) degree, and coefficients of polynomial (in fact constant) size. According to Lemma 1, this shows that Gaussian elimination runs in polynomial time in the weak model.

# 4 Small points in semi-algebraic sets

The proof of Theorem 4 relies on the following technical result.

**Theorem 6** *Let $\mathcal{S} \subset \mathbb{R}^p$ be defined by a system of inequalities*

$$P_i(x) > 0, \;\; i = 1 \dots N$$

*with $P_i \in \mathbb{Z}[X_1, \dots, X_p]$.*

*Let $D$ be the maximum degree of the $P_i$'s, and $S$ the maximum size of their coefficients. If $\mathcal{S} \neq \emptyset$, there exists a rational point $x \in \mathcal{S}$ of size $s(x) \leq aSD^b$. The constant $a$ and $b$ depend only on $p$.*

This result states that if $\mathcal{S} \neq \emptyset$ is defined by “small” polynomials, it contains a “small” rational point. Note that the number of inequations does not come into play in this bound.

Before proving Theorem 6, we present some useful results from computer algebra. These tools are necessary for the proof of Lemma 5, but the rest of this section can be read independently.

## 4.1 Basic tools

The following bound is well known.

**Lemma 3 (Cauchy, 1829)** *Let* $P(x) = \sum_{i=0}^{D} a_i x^i$ $(a_D \neq 0)$ *be a complex polynomial, and*

$$m = \inf\{|a_i|, i = 0 \ldots D, a_i \neq 0\}, M = \sup\{|a_i|, i = 0 \ldots D\}.$$

*If* $\alpha \in \boldsymbol{C}$, $\alpha \neq 0$ *is a root of* $P$ *then*

$$\frac{m}{m+M} < |\alpha| < \frac{m+M}{m}.$$

The sign vector of a sequence of polynomials $P_1, \ldots, P_N \in \mathbb{R}[X]$ at a point $x \in \mathbb{R}$ is the vector $(\epsilon(P_i(x)))_{1 \leq i \leq N} \in \{-1, 0, 1\}^N$, with, by definition, $\epsilon(y) = 1$ if $y > 0$, 0 if $y = 0$, -1 if $y < 0$. Usually, a real root $\alpha$ of a polynomial $P \in \mathbb{R}[X]$ is coded by an interval $]a, b[$ containing $\alpha$ and no other root of $P$. A purely symbolic coding method was proposed in [7]: the roots of $P$ are characterized by the sign vectors of the derivatives of $P$. More precisely, if $d(P) \leq D$, $\alpha$ is characterized by the sign vector

$$\epsilon(\alpha) = (\epsilon(P^{(i)}(\alpha))_{1 \leq i \leq D} \in \{-1, 0, 1\}^D.$$

According to Thom's lemma, a root is uniquely determined by its sign vector: if $P(\alpha) = P(\beta) = 0$ and $\epsilon(\alpha) = \epsilon(\beta)$, then $\alpha = \beta$. It is moreover possible to compare two roots $\alpha$ and $\beta$ by comparing only their sign vectors. More formally, there exists a comparison function $c_D : \{-1, 0, 1\}^D \times \{-1, 0, 1\}^D \rightarrow \{-1, 0, 1\}$ independent of $P$ such that $c_D(\epsilon(\alpha), \epsilon(\beta)) = -1$ if $\alpha < \beta$, 0 if $\alpha = \beta$, 1 if $\alpha > \beta$. $c_D$ can be actually computed with a very simple algorithm (see [7] for more details).

Given $a \in \mathbb{R}^{D+1}$, let $P_a$ be the polynomial$P_a(X) = \sum_{i=0}^{D} a_i X^i$, and $r(P_a)$ the number of distinct real roots of $P_a$. The following key result is a special case of the "quantifier elimination subproblem" of [22]. Similar bounds were also provided in [9].

**Lemma 4 (Renegar)** *Let* $N_1, \ldots, N_M$ *be positive integers, and* $\epsilon_1, \ldots, \epsilon_M \in \{-1, 0, 1\}^N$ *sign vectors. The set* $S_{a,b} \subset \mathbb{R}^{N(D+1)} \times \mathbb{R}^{D+1}$ *of coefficients* $(a, b) = (a_1, \ldots, a_N, b) \in \mathbb{R}^{D+1} \times \mathbb{R}^{D+1} \times \ldots \times \mathbb{R}^{D+1}$ *satisfying the following two properties is semi-algebraic:*

*(i)* $r(P_b) = N_1 + \ldots + N_M$;

*(ii)* $N_j$ *is the number of distinct real zeros of* $P_b$ *at which the sign vector of* $(P_{a_i})_{1 \leq i \leq N}$ *is* $\epsilon_j$, *for all* $j = 1, \ldots, M$.

*Moreover,* $S_{a,b}$ *can be defined by polynomial (in)equalities of degree* $(ND)^{O(1)}$ *with coefficients of size* $(ND)^{O(1)}$.

## 4.2 Proof of theorem 6

Given a polynomial $P \in \mathbb{R}[X]$, we denote by $\alpha_\epsilon$ its root of sign vector $\epsilon$ – if there is such a root. In this section, the following notation will be used: $]a, b[$ denotes the interval $]\min(a, b), \max(a, b)[$.

**Lemma 5** *For $D \geq 1$, let $S_{\epsilon,\epsilon'} \subset \mathbb{R}^{D+1} \times \mathbb{R}^{D+1}$ be the set of coefficients $(a, b)$ such that*

$$P_b(x) > 0 \text{ for } x \in ]\alpha_\epsilon, \alpha_{\epsilon'}[ \tag{1}$$

*where $\alpha$ is the sequence of roots of $P_a$. $S_{\epsilon,\epsilon'}$ is a semi-algebraic set defined by polynomials of degree $D^{O(1)}$ whose coefficients are of size $D^{O(1)}$.*

*Proof.* We can assume without loss of generality that $c_D(\epsilon, \epsilon') = -1$. This implies that $\alpha_\epsilon < \alpha_{\epsilon'}$ whenever these roots exist. Lemma 4 will be applied several times to $P_a$, $P_b$, $P_aP_b$ and their derivatives. We call a semi-algebraic "small" if it is defined by polynomials of degree $D^{O(1)}$ whose coefficients are of size $D^{O(1)}$.

Let $B_{\epsilon,\epsilon'}$ be the set of coefficients $(a, b) \in \mathbb{R}^{2(D+1)}$ such that $P_b \neq 0$ on $]\alpha_\epsilon, \alpha_{\epsilon'}[$. In order to compare the roots of $P_a$ and $P_b$, we consider the polynomial $P_aP_b$ (whose zeros are the roots of $P_a$ and $P_b$). According to Lemma 4 applied to $P_a$, $P_aP_b$ and their derivatives, $B_{\epsilon,\epsilon'}$ is a small semi-algebraic set, since $(a, b) \in B_{\epsilon,\epsilon'}$ if

1. $r(P_aP_b) = r \geq 2$;
2. The sign vectors of the sequence of $2D + 1$ polynomials

   $$\left(\left((P_aP_b)^{(i)}\right)_{1 \leq i \leq D}, P_a, \left(P_a^{(i)}\right)_{1 \leq i \leq D}\right)$$

   at the zeros of $P_aP_b$ are

   $$(\sigma_i, \alpha_i, \epsilon_i) \in \{-1, 0, 1\}^D \times \{-1, 0, 1\} \times \{-1, 0, 1\}^D$$

   for $1 \leq i \leq r$.
3. $c_{2D}(\sigma_i, \sigma_j) = -1$ for all $i < j$, and there exists $l < r$ such that $\alpha_l = \alpha_{l+1} = 0$, $\epsilon_l = \epsilon$ and $\epsilon_{l+1} = \epsilon'$.

If $P_b \neq 0$ on $]\alpha_\epsilon, \alpha_{\epsilon'}[$, $P_b$ is either positive or negative on this interval. More precisely, $P_b$ will take the sign of its first non-zero derivative in $\alpha_\epsilon$. We thus define $C_\epsilon$ as the set of coefficients $(a, b)$ such that the first non-zero derivative of $P_b$ in $\alpha_\epsilon$ is positive. According to Lemma 4 applied to $P_a$, $P_b$ and their derivatives, $C_\epsilon$ is a small semi-algebraic set, since $(a, b) \in C_\epsilon$ if

1. $r(P_a) = r \geq 1$;
2. The sign vectors of the sequence of $2D + 1$ polynomials $((P_a^{(i)})_{1 \leq i \leq D}, (P_b^{(i)})_{0 \leq i \leq D})$ at the zeros of $P_a$ are $(\epsilon_1, \sigma_1), \ldots, (\epsilon_r, \sigma_r)$;

3. There exists $j$ such that $\epsilon_j = \epsilon$ and the first non-zero component of $\sigma_j$ is positive.

The claim now follows from the relation $S_{\epsilon,\epsilon'} = B_{\epsilon,\epsilon'} \cap C_\epsilon$. □

**Lemma 6** *Given* $a \in \mathbb{R}^{N(D+1)}$, *let* $(P_{a_i})_{1\leq i\leq N}$ *be the sequence of polynomials such that* $P_{a_i} = \sum_{j=0}^{D} a_{ij}X^j$. *Let*

$$S_a = \{x \in \mathbb{R}, P_{a_1}(x) > 0 \wedge P_{a_2}(x) > 0 \wedge \ldots \wedge P_{a_N}(x) > 0\}.$$

*The set of coefficients*

$$\mathcal{C} = \{a \in \mathbb{R}^{N(D+1)}, \mathcal{S}_a \neq \emptyset\}$$

*is a semi-algebraic subset of* $\mathbb{R}^{N(D+1)}$ *defined by polynomials of degree* $D^{O(1)}$ *whose coefficients are of size* $D^{O(1)}$. *No more than* $3(D+1)$ *of the* $N(D+1)$ *variables* $a_{ij}$*'s can occur in a given polynomial.*

*Proof.* If $\mathcal{S}_a \neq \emptyset$ for a given $a \in \mathbb{R}^{N(D+1)}$, there exists a maximal non-empty interval $]\alpha, \beta[\subset \mathcal{S}_a$. If $\alpha$ and $\beta$ are finite, they are roots of two polynomials $P_{a_i}$ and $P_{a_j}$ (possibly, $i = j$). Hence the set $\mathcal{C}_1$ of vectors $(a, b) \in \mathcal{C}$ corresponding to the case where both $\alpha$ and $\beta$ are finite satisfies:

$$\mathcal{C}_1 = \bigcup_{i,j,\epsilon,\epsilon'} \bigcap_{l} \mathcal{C}_{ij\epsilon\epsilon' l}$$

where

$$\mathcal{C}_{ij\epsilon\epsilon' l} = \left\{a \in \mathbb{R}^{N(D+1)}, P_{a_l} > 0 \text{ on } ]\alpha_{ij\epsilon}, \alpha_{ij\epsilon'}[\right\}$$

and $\alpha_{ij\eta}$ is the root of $P_{a_i}P_{a_j}$ of sign vector $\eta$. The conclusion of the lemma holds for each $\mathcal{C}_{ij\epsilon\epsilon' l}$ according to Lemma 5, hence it holds also for $\mathcal{C}_1$. The cases where $\alpha$ or $\beta$ are infinite can be handled with a straightforward adaptation of Lemma 5. □

*Proof of Theorem 6.* By induction on $D$. For $D = 1$, as was noticed in the proof of lemma 6, there exists a maximal non-empty interval $]\alpha, \beta[\subset \mathcal{S}$ whose endpoints, when finite, are roots of two polynomials $P_i$ and $P_j$. It follows from the standard bound on the roots of polynomials that $\alpha$ and $\beta$ cannot be too large (if they are finite). $\beta - \alpha$ cannot be too small since it is the root of a polynomial of degree $D^{O(1)}$ with integer coefficients of size $O(SD^{O(1)})$ [19]. Hence there are small rational points in $]\alpha, \beta[$.

Assume now that the result holds in dimension $p - 1$. In order to prove that it holds for $\mathcal{S} \subset \mathbb{R}^p = \mathbb{R}^{p-1} \times \mathbb{R}$, $S \neq \emptyset$ we consider the projection $\mathcal{S}'$ of $\mathcal{S}$ on $\mathbb{R}^{p-1}$. It will be shown that $\mathcal{S}'$ is semi-algebraic (this is just the Tarski-Seidenberg principle) and described by small polynomials. Hence $S'$ contains a small point $x_0$ by induction hypothesis. The polynomials $P_i(x_0, x_p)$ have small coefficients, hence there exists a small $x_p$ such that $(x_0, x_p) \in \mathcal{S}$. We now fill in the missing details.

Let us consider each $P_i$ as a polynomial in $x_p$ only: one can write

$$P_i(x) = \sum_{j=0}^{D} a_{ij}(x')x_p^j$$

with $x' = (x_1, x_2, \ldots, x_{p-1})$. In the notations of Lemma 6, $x' \in \mathcal{S}'$ if and only if $a(x') = (a_{ij}(x'))_{i,j} \in \mathcal{C}$. Let $c$ and $d$ be the constants such that the degrees of the polynomials describing $\mathcal{C}$ and the size of their coefficients are smaller than $cD^d$. The $a_{ij}$'s are polynomials of degree at most $D$, therefore $S'$ can be described by polynomials of degree at most $cD^{d+1}$. A simple calculation also shows that the size of their coefficients is bounded by $c'SD^{d'}$ for some constants $c'$ and $d'$. The decomposition of $S'$ in basic semi-algebraic sets is of the form

$$S' = \bigcup_{i=1}^{m} S'_i$$

with

$$S'_i = \{x' \in \mathbb{R}^{p-1}; Q_{i1} > 0 \wedge \ldots \wedge Q_{ij_i} > 0 \wedge R_{i1} = 0 \wedge \ldots \wedge R_{ik_i} = 0\}.$$

$\mathcal{S}'$ is open since $\mathcal{S}$ is open, therefore $k_i = 0$ for at least one of the $\mathcal{S}'_i$'s, say, $\mathcal{S}'_1$. By induction hypothesis, there exist constants $a'$ and $b'$ and a point $x_0 \in \mathcal{S}'_1$ of size

$$s(x_0) \leq a'(c'SD^{d'})(cD^{d+1})^{b'} :$$

there are clearly constants $a$ and $b$ such that $s(x_0) \leq aSD^b$. A similar bound can be obtained for $x_p$ by substituting $x_0$ to $x'$ in the polynomials $P_i(x', x_p)$, clearing the denominators and applying Theorem 6 in dimension 1. $\square$

# 5 Proof of the main result

In this section, we complete the proof of Theorem 4. From now on, it is assumed without loss of generality that all comparisons are of the form "$x > 0$" where $x$ is a variable. The proof will first be made under the following hypothesis, and then in the general case:

(H) Whenever a test $x > 0$ is performed, $x$ is not equal to zero.

Two solutions will be provided for the general case. In this section, we shall see that (H) can be assumed without loss of generality. It is shown in section 6 that equality cases can be handled with symbolic computation techniques. Note that (H) holds if the $\alpha_i$'s are algebraically independent.

**Theorem 7** *If a language $L \subset \{0,1\}^*$ can be recognized in polynomial time under hypothesis* (H)*, then $L \in$ P/poly.*

*Proof.* The main idea is to replace the constants $\alpha_1, \ldots, \alpha_p$ by rational approximations $\overline{\alpha_1}, \ldots, \overline{\alpha_p}$. In accordance with the definition of advice functions, these constants must not fully depend on the input $u$, but only on its length $n = |u|$. The program is then run with these new constants: at time $t$, the content of the variable $x$ is now $\overline{x}(t)$ instead of $x(t)$. Two points need to be checked:

1. The outcome of the computation is the same.

2. The computation remains polynomial-time.

The first condition will be satisfied if for any variable $x$ and time $t$, $x(t) < 0 \Rightarrow \overline{x}(t) < 0$ and $x(t) > 0 \Rightarrow \overline{x}(t) > 0$. As a matter of fact, these requirements ensure that the results of the tests (including the final test on $R$) will be the same in both programs. Let $T(n) = O(n^l)$ be the worst-case running time for inputs of length $n$. According to Lemma 1, we obtain a system of polynomial equations on $\overline{\alpha} = (\overline{\alpha_1}, \dots, \overline{\alpha_p})$ of the type

$$\mathcal{E} = \{P_{t,u}(\overline{\alpha}) > 0\}_{t \leq T(n), |u|=n} .$$

An equation $P_{t,u} > 0$ occurs in $\mathcal{E}$ if the instruction executed at time $t$ on input $u$ is a test on a variable $x$, and in this case

$$P_{t,u} = \mathrm{sign}(x(t)).R_{\mathcal{N},x}.S_{\mathcal{N},x},$$

$R_{\mathcal{N},x}$ and $S_{\mathcal{N},x}$ being the polynomials such that $Q_{\mathcal{N},x} = R_{\mathcal{N},x}/S_{\mathcal{N},x}$ (the rational fractions $Q_{\mathcal{N},x}$ are defined in section 3). $\mathcal{E}$ is a satisfiable system, since $\alpha$ is obviously a solution. Theorem 6 and Lemma 1 will now be invoked to show that $\alpha$ can be replaced by a rational approximation which needs not be too accurate. This will ensure the second condition.

According to Lemma 1, the $P_{t,u}$'s are of polynomial degree and their coefficients of polynomial size. It follows from Theorem 6 that $\mathcal{E}$ has a rational solution $\overline{\alpha}$ of polynomial size. The approximations $\overline{x}(t)$ can thus be computed in polynomial time. $\Box$

Note that $\mathcal{E}$ can contain an exponential number of inequations. It is therefore crucial that this number does not come into play in the bound of Theorem 6. If for example $D$ were the sum of the degrees instead of the maximum degree, our proof would fail. A result of this type was provided recently in [23].

We now show how to get rid of equality cases in the general case; this shows that (H) is in fact not restrictive. This solution is based on the following obvious remark: if a test "$x > 0$" is performed during the execution of a RTM program $\mathcal{P}$, and if $x \neq 0$, there exists $\epsilon > 0$ such that either $x > \epsilon$ or $x < -\epsilon$. For a fixed $n$, only a finite number of tests are performed during the computations on all inputs of length $n$, hence we can choose the same $\epsilon$ for all these tests. For each $n$, we define a new system $\mathcal{E}''$ of inequations on $\overline{\alpha_1}, \dots, \overline{\alpha_p}$ and on a new variable $\overline{\epsilon}$. $\mathcal{E}''$ is obtained from $\mathcal{E}$ by the following operations:

1. add the inequation $\overline{\epsilon} > 0$;

2. replace the inequation associated to a test "$x > 0$" by the inequation associated to the test:

   (a) $x - \overline{\epsilon} > 0$ if it turns out that $x > 0$;

   (b) $x - \overline{\epsilon} < 0$ if it turns out that $x \leq 0$.

$\mathcal{E}''$ is satisfiable since $(\alpha_1, \ldots, \alpha_p, \epsilon)$ is a solution, hence there exists a "small" solution $(\overline{\alpha_1}, \ldots, \overline{\alpha_p}, \overline{\epsilon})$ according to Theorem 6. The program obtained by replacing the $\alpha_i$'s by the $\overline{\alpha}_i$'s and the tests "$x > 0$" by "$x > \overline{\epsilon}$" runs in polynomial time and is equivalent to $\mathcal{P}$.

The symbolic solution to the problem of equality cases is described in detail in the next section. For each test "$x > 0$", we test whether $x$ is equal to 0 with the algorithm of Theorem 8. If it turns out that $x = 0$, then of course we know that the answer to the original test is negative. If it turns out that $x \neq 0$, then we can use small rational constants $\overline{\alpha_i}$ as above.

# 6 Order-free models

In this section, we consider computation models for which the order relation is not available. Therefore the only authorized tests are equality tests. We first give a simulation result for the weak BSS model at the beginning of section 6.1, and extend it to the full (order free) BSS model by probabilistic methods. The corresponding complexity classes are denoted by the "=" superscript.

Section 6.2 deals with a more conventional model of computation, namely, arithmetic circuits over the integers. We consider families $(C_n)_{n \in \mathbb{N}}$ of arithmetic circuits of polynomial size working with parameters of exponential size. An arbitrary boolean function of $n$ variables can be described by a $2^n$-digit number, and can thus be encoded in the parameters of a circuit. But is there an encoding scheme that allows to evaluate this function efficiently with arithmetic circuits ? In fact this is not the case: we show that up to a polynomial increase in size, these circuits are not more powerful than boolean circuits.

## 6.1 Real Turing machines

In order to perform a test $x \stackrel{?}{=} 0$, we will explicitely compute the polynomials $R$ and $S$ such that $x = R(\alpha)/S(\alpha)$. This can be done in polynomial time according to Lemma 1 (recall that a polynomial of degree $d$ in $p$ variables has only $O(d^p)$ monomials). We need to determine whether $R$ belongs to the ideal $\mathcal{I} = \{P \in \mathbb{Z}[X], P(\alpha) = 0\}$. This problem can be solved by symbolic computation techniques because the ideals of $\mathbb{Z}[X]$ are finitely generated, *i.e.*,

$$\mathcal{I} = \left\{\sum_{i=1}^{m} Q_i G_i, Q_1, \ldots, Q_m \in \mathbb{Z}[X]\right\}$$

for some finite system of generators

$$\mathcal{G} = (G_1, \ldots, G_m) \in \mathbb{Z}[X]^m.$$

This follows from Hilbert's theorem on nœtherian rings (see e.g. [18], chapter 6). Hence we just have to determine if $R$ belongs to the ideal generated by $\mathcal{G}$. One could use Gröbner basis techniques to solve this problem, but this is in fact not necessary. Bruno Poizat (personal communication) noticed that the primitive element theorem [18] suffices. The argument is as follows.

Let $k$ be the transcendence degree over $\mathbb{Q}$ of the field $K = \mathbb{Q}[\alpha_1, \ldots, \alpha_p]$. We can assume without loss of generality that $\alpha_1, \ldots, \alpha_k$ is a transcendence base of $K$. The other constants $\alpha_{k+1}, \ldots, \alpha_p$ are algebraic over $\mathbb{Q}[\alpha_1, \ldots, \alpha_k]$. It follows from the primitive element theorem that

$$K = \mathbb{Q}[\alpha_1, \ldots, \alpha_k][\alpha_{k+1}, \ldots, \alpha_p] = \mathbb{Q}[\alpha_1, \ldots, \alpha_k][\beta]$$

where the primitive element $\beta \in \mathbb{R}$ is algebraic over $\mathbb{Q}[\alpha_1, \ldots, \alpha_k]$. Hence there are rational fractions $Q_j$ such that

$$\alpha_j = Q_j(\alpha_1, \ldots, \alpha_k, \beta),\ j = k+1, \ldots, p \tag{2}$$

and the content of register $x$ is actually a rational fraction $x = R(\alpha_1, \ldots, \alpha_k, \beta)$. This property yields a very convenient way of testing whether $x$ is equal to zero. We just have to make sure that as a polynomial in $\beta$, the numerator $P_1$ of the fraction $R = P_1/P_2$ is a multiple of the minimum polynomial $M$ of $\beta$ over $\mathbb{Q}[\alpha_1, \ldots, \alpha_k]$. Checking this condition is not difficult: perform the division of $P_1$ by $M$, and since $\alpha_1, \ldots, \alpha_k$ are algebraically independent, the remainder $R = P_1 \bmod M$ is equal to zero if its coefficients (which are in $\mathbb{Q}[\alpha_1, \ldots, \alpha_k]$) are identically equal to zero.

We can assume that $M$ and the $Q_j$'s are "hardwired" in the program, since they are independent of the input. The whole computation therefore takes polynomial time. We have proved the following result:

**Theorem 8** $\mathrm{BP}(\mathrm{P}^{=}_{W}) = \mathrm{P}$.

Note that the symbolic technique developed in this section is more powerful than the "numerical" technique used in the previous section to deal with equality cases. Indeed, this latter technique would only show that $\mathrm{BP}(\mathrm{P}^{=}_{W}) \subset \mathrm{P/poly}$ instead of the stronger result $\mathrm{BP}(\mathrm{P}^{=}_{W}) = \mathrm{P}$. In the full BSS model, we are only able to prove the following weaker result.

**Theorem 9** $\mathrm{BP}(\mathrm{P}^{=}_{R}) \subset \mathrm{BPP}$.

Recall that BPP is the class of (boolean) languages that can be recognized in polynomial time by a probabilistic Turing machine with a bounded probability of error (see, e.g., [1] for more details). Before moving to the proof of Theorem 9, we present two lemmas which are often used in the analysis of randomized algorithms (see [15] for a proof of the first one).

**Lemma 7 (Schwartz)** *Let $P(x_1, \ldots, x_p)$ be a polynomial of degree $d$. Choose the $x_i$'s independently from the uniform distribution on $\{0, 1, \ldots, N-1\}$.If $P \not\equiv 0$, then*

$$\Pr\{P(x_1, \ldots, x_p) = 0\} \leq dp/N.$$

**Lemma 8** *Let $x \neq 0$ be an integer smaller than $2^{2^n}$. For any constant $c > 1$ there is a constant $d > 0$ such that the number of integers smaller than $2^{cn}$ not dividing $x$ is at least $d2^{cn}/n$.*

*Proof.* By the law of prime numbers there are at least $\Omega(2^{cn}/cn)$ primes smaller than $2^{cn}$. However $x$ cannot have more than $2^n$ prime divisors. □

An application of this lemma in a very similar context can be found in [24].

**Corollary 2** *Let $x \neq 0$ be an integer smaller than $2^{2^n}$. Choose $m_1, \ldots, m_v$ independently from the uniform distribution on $\{0, 1, \ldots, 2^{cn} - 1\}$, for some constant $c > 1$. There is a constant $C > 0$ such that for every $\epsilon > 0$ and any $v > Cn \log \frac{1}{\epsilon}$,*

$$\Pr\{x \bmod m_1 = x \bmod m_2 = \ldots = x \bmod m_v = 0\} \leq \epsilon.$$

*Proof.* By Lemma 8, the probability to be bounded is smaller than $(1 - d/n)^v$. □

*Proof of Theorem 9.* The algorithm of Theorem 8 no longer runs in polynomial time, since rational fractions of exponential degree can be created in polynomial time (by iterated multiplication). As a first modification, we can compute the pairs $(P_1 \bmod M, P_2 \bmod M)$ instead of computing the rational fractions $R = P_1/P_2$ ($P_1, P_2$ and $M$ are still viewed as elements of $\mathbb{Q}[\alpha_1, \ldots, \alpha_k][X]$). Hence we now work with polynomials of bounded degree in $\beta$, but still face an exponential growth problem for their coefficients. These coefficients can be computed with a polynomial number of arithmetic operations in $\mathbb{Q}[\alpha_1, \ldots, \alpha_k]$. In order to decide if such a polynomial $P(\alpha_1, \ldots, \alpha_k)$ is equal to zero, we will not compute its coefficients explicitly. Instead, we replace $\alpha_1, \ldots, \alpha_k$ by randomly chosen integers $n_1, \ldots, n_k$ and perform the same sequence of arithmetic operations. If it turns out that $P(n_1, \ldots, n_k) \neq 0$, then certainly $P \neq 0$. If it turns out that $P(n_1, \ldots, n_k) = 0$, then $P \equiv 0$ with high probability according to Lemma 7 (detailed analysis below). Unfortunately, this new sequence of arithmetic operations cannot be performed in polynomial time either, since we could create integers of exponential size. However, this problem can be solved with high probability by performing the operations modulo $v$ randomly drawn integers $m_1, \ldots, m_v$ (Corollary 2).

Let us now show that in order to have a probability of error smaller than, say, 1/4, the running time of this probabilistic algorithm is indeed polynomial. Since the original program runs in polynomial time, it suffices to have a probability of error smaller than $1/n^{O(1)}$ for each test. For any given test, the probability of error is bounded by the sum of errors for the two steps:

1. application of Schwarz's lemma.

2. application of Corollary 2.

Since the polynomials in $\alpha_1, \ldots, \alpha_k$ occurring in the computations of the pairs ($P_1 \bmod M, P_2 \bmod M$) are obtained by a polynomial number of arithmetic operations, their degrees are smaller than $2^{n^{O(1)}}$. It follows from Schwarz's lemma that the randomly drawn integral points $(n_1, \ldots, n_k)$ at which these polynomials are evaluated can be taken of polynomial size.

It is also clear from Corollary 2 that in order to achieve an error probability smaller than $1/n^{O(1)}$, drawing a polynomial number of integers $m_i$ of polynomial size is sufficient.

Let us now summarize the overall algorithm. It is easily verified that the same random integers $n_1, \ldots, n_k, m_1, \ldots, m_v$ can be used for all tests. Hence the fist step of our algorithm will be:

1. On an input $u \in \{0,1\}^n$, choose at random $k$ integers $n_1, \ldots, n_k$ in $\{0, 1, \ldots, 2^{n^{O(1)}} - 1\}$ and $v = n^{O(1)}$ integers $m_1, \ldots, m_v$ in $\{0, 1, \ldots, 2^{n^{O(1)}} - 1\}$.

Each register or constant $x$ of the real Turing machine will be represented by a sequence of rational fractions $(R_i^x)_{1 \leq i \leq v}$, where $R_i^x = P_i^x / Q_i^x$ and $P_i^x, Q_i^x \in \mathbb{Q}[X]$. These fractions are initialized as follows:

2. $R_i^{\alpha_j} = n_j \bmod m_i$ for $1 \leq j \leq k$; $R_i^{\alpha_j}(X) = Q_j(n_1, \ldots, n_k, X)$ for $k+1 \leq j \leq p$. The coefficients of $R_i^{\alpha_j}$ are to be computed modulo $m_i$.

3. An operation $x := y$ op $z$ is simulated by the following sequence of operations:

$$R_i^x := (R_i^y \text{ op } R_i^z) \bmod M(n_1, \ldots, n_k, X)$$

   for $i = 1, \ldots, v$; the mod operator specifies that the numerator and denominator of $R_i^x$ are to be computed modulo $M$. Moreover, the coefficients of $R_i^x$ are to be computed modulo $m_i$.

4. For a test $x \stackrel{?}{=} 0$, a positive answer is provided if and only if $P_i^x = 0$ for every $i = 1, \ldots, v$.

At step 2, we assume without loss of generality that $\deg_X(Q_j) < \deg_X(M)$. This explains why, contrary to step 3, we do not have to compute a remainder modulo $M$. $\square$

## 6.2 Arithmetic circuits over the integers

There is no obvious way to apply probabilistic techniques in the full BSS model with order. One natural approach would be to use Theorem 6 to replace for each input size $n$ the real constants $\alpha$ by rational constants $\overline{\alpha}$ of exponential size. This shows that real Turing machines can be simulated by polynomial-size arithmetic circuits over the rationals (or equivalently, over the integers). Now one would like to simulate these arithmetic circuits by boolean circuits. Of course, the problem is that the modular computations of section 6.1 completely break down if order is available. It is nonetheless possible to say something about arithmetic circuits by going back to the order-free model.

**Theorem 10** *Let $(C_n)_{n \in \mathbb{N}}$ be a family of (acyclic) arithmetic circuits of size polynomial in $n$. We assume that $C_n$ has $n$ boolean inputs, one boolean output and that the bit size of each of its parameter is smaller than $2^n$. These circuits are made of three types of gates:*

- *Addition gates of the form $z := x + y$;*
- *Multiplication gates of the form $z := xy$;*
- *Selection gates of the form $z$:=(if $x = 0$ then $y_1$ else $y_2$)*

*where $z$ is the output of the gate and $x, y, y_1, y_2$ are outputs of other gates, parameters of the circuit, or boolean inputs. Under the preceding assumptions, $(C_n)_{n\in\mathbb{N}}$ can be simulated by a family of boolean circuits of polynomial size.*

*Proof.* The idea is to perform the computations of $C_n$ modulo several random integers of polynomial size, as in the proof of Theorem 9. If $S(n)$ is the size of $C_n$, the integers occurring in a computation of this circuit are smaller than $2^{2^{n+S(n)}}$, i.e., they remain of singly exponential size. It follows from Corollary 2 that $C_n$ can be simulated correctly with high probability using a polynomial number of random integers $m_1, \ldots, m_v$. In order to show that $C_n$ can be simulated by a small boolean circuit $B_n$, we seek a sequence $m_1, \ldots, m_v$ that provides the correct answer for all inputs of $C_n$. The existence of such a sequence (called a "good advice") follows from a standard counting argument: let $q(n)$ be the (polynomial) number $v$ of random trials that guarantees an error probability smaller than $2^{-2n}$. For a fixed input $u \in \{0,1\}^n$, there are by hypothesis at most $2^{q(n)-2n}$ advice $m_1, \ldots, m_{q(n)}$ that are bad for this input. Hence there are at most $2^{n+q(n)-2n}$ advice that are bad for some input. This shows that there are at least $2^{q(n)}.(1-2^{-n})$ good advice. □

We conclude with a list of a few possible generalizations of Theorem 10. A solution to these problems will hopefully result in a better understanding of the Blum-Shub-Smale model.

**Problem 1** *What happens if:*

- *comparisons of the form $x < y$ are allowed ?*
- *the size of the parameters of $C_n$ can be as large as $2^{2^n}$ ?*
- *the parameters of $C_n$ are unbounded ?*

# 7 A separation theorem

In this section, we show that $\mathrm{NP}_R$ is strictly included in $\mathrm{EXP}_W$, the class of real languages recognized in weak exponential time by real Turing machines. The large inclusion was established in [12]. The proof answers a question of [13]: is there a boolean language which is not in $\mathrm{BP}(\mathrm{P}_R)$ ? In fact, we show that some boolean languages are not in $\mathrm{BP}(\mathrm{NP}_R)$ (and *a fortiori* not in $\mathrm{BP}(\mathrm{P}_R)$).

We need two results from real algebraic geometry. The following bound on the number of satisfiable sign conditions for a set of multivariate polynomials is due to Milnor [21]; see also [22] for an "elementary" proof.

**Theorem 11 (Milnor)** *Let $\{g_1, \ldots, g_m\}$ be a set of of $m$ polynomials in $n$ variables. A sign condition $\epsilon \in \{-1, 0, 1\}^m$ is said to be consistent if there exists some $x \in \mathbb{R}^n$ such that $\epsilon_i$ is the sign of $g_i(x)$, for every $i = 1, ..., m$.*

*If the $g_i$'s are of degree at most $d$, the number of consistent sign conditions is $(md)^{O(n)}$.*

We also need the following special case of Renegar's quantifier elimination results [22].

**Theorem 12 (Renegar)** *Let $F$ be an existential formula of the first-order therory of the reals: $F$ is of the form*

$$\exists y \in \mathbb{R}^N G(x,u)$$

*where $G$ is a quantifier-free formula, and the free variable $u$ is in $\mathbb{R}^l$. Let $m$ be a bound on the number of polynomials occuring in $F$, and $d$ a bound on their degrees.*

*Then $F(u)$ is equivalent to a quantifier-free formula $H(u)$. The number of polynomials occuring in $H$ is at most $(md)^{O(lN)}$, and their degrees at most $(md)^{O(N)}$.*

We begin with a few general remarks.

**Lemma 9** *Assume that $L \subset \mathbb{R}^\infty$ can be recognized in polynomial time by a real Turing machine $\mathcal{M}$. Then $L$ can be recognized by a family $(C_n)_{n\in\mathbb{N}}$ of polynomial-size arithmetic circuits over the reals. Moreover, $C_n$ uses the same real constants as $\mathcal{M}$, and $C_n$ can be constructed in polynomial time by an ordinary Turing machine.*

The proof of this result will be omitted since it is really the same as in the boolean case (i.e., simulation of Turing machines by boolean circuits [1]). In fact, Lemma 9 can be generalized to models of computation over arbitrary structures (see [13]).

Now, let $\mathcal{M}$ be a polynomial time non-deterministic real Turing machine. We can assume without loss of generality that for every input $x \in \mathbb{R}^n$, $\mathcal{M}$ uses exactly $g(n)$ guesses, where $g$ is a polynomially bounded function. It follows immediately from Lemma 9 that there exists a sequence $(C_n)_{n>0}$ of arithmetic circuits over the reals satisfying the following properties:

- The output of $C_n$ on any input $(x,y) \in \mathbb{R}^n \times \mathbb{R}^{g(n)}$ and the output of $\mathcal{M}$ on input $x$ and guess $y$ are identical.
- The size $s_n$ of $C_n$ is polynomially bounded.
- The real constants $\alpha = (\alpha_1, \ldots, \alpha_p)$ of $C_n$ and $\mathcal{M}$ are identical.

Note that for every $x \in \mathbb{R}^n$, the property

$$\exists y \in \mathbb{R}^{g(n)} C_n(x,y) = 1$$

is equivalent to an existential *formula* of the form

$$\exists Y \in \mathbb{R}^{h(n)} F_n(x,Y). \tag{3}$$

The quantifier-free formula $F_n$ is of size $O(s_n)$ and $h(n)$ is bounded by $g(n)+s(n) \leq 2s(n)$. This equivalent formula is simply obtained by adding new quantified variables $Y_i$ representing the outputs of all internal gates of $C_n$. This transformation can also be performed in arbitrary structures [13]. The following result of [12] follows immediately.

**Theorem 13** $\mathrm{NP}_R = \mathrm{NP}_W$.

*Proof.* In order to to simulate $\mathcal{M}$ in non-deterministic weak polynomial time on an input $x \in \mathbb{R}^n$, construct $F_n(x)$ and check that this formula is satisfiable by guessing $Y \in \mathbb{R}^{h(n)}$. $\square$

We are now ready to prove the main result of this section.

**Theorem 14** $\mathrm{NP}_R$ *is strictly included in* $\mathrm{EXP}_W$.

*Proof.* It is shown in [12] that $\mathrm{NP}_R \subseteq \mathrm{EXP}_W$. Hence we just have to prove that $\mathrm{NP}_R \neq \mathrm{EXP}_W$. These two sets are different because their boolean parts are different. The boolean part of $\mathrm{EXP}_W$ is $2^{\{0,1\}^*}$, i.e., every boolean language is in $\mathrm{BP}(\mathrm{EXP}_W)$. We shall show by a counting argument that some boolean languages are not in $\mathrm{BP}(\mathrm{NP}_R)$.

Let $\mathcal{M}$ be a polynomial time non-deterministic real Turing machine. As in the preceding discussion, we assume that for every input $x \in \mathbb{R}^n$, $\mathcal{M}$ uses exactly $g(n)$ guesses. Let $(C_n^\alpha)_{n>0}$ be the family of arithmetic circuits constructed above. We use the notation $C_n^\alpha$ instead of $C_n$ to emphasize the role of real constants. If the constants $\alpha$ are replaced by new constants $\beta$, we obtain a new circuit $C_n^\beta$. The two circuits $C_n^\alpha$ and $C_n^\beta$ recognize different sets in general. However, when the input $x$ is boolean and $y$ is interpreted as a real guess, we shall see that it is possible to bound the number of "truly different" real constants. To make this precise, we say that $\alpha \equiv \beta$ if for every input $x \in \{0,1\}^n$,

$$\exists y \in \mathbb{R}^{g(n)} C_n^\alpha(x,y) = 1 \Leftrightarrow \exists y \in \mathbb{R}^{g(n)} C_n^\beta(x,y) = 1.$$

We now show that there is a universal constant $a$ such that the number $E$ of equivalences classes of $\equiv$ satisfies

$$E \leq 2^{ap(n+p)s_n \log s_n}. \tag{4}$$

Let us apply Theorem 12 to formula (3), with $N \leq 2s(n)$, $m \leq s(n)$, $l = n + p$ and $d = 2$. We obtain an equivalent quantifier-free formula $H_n^\alpha(x)$ in $n + p$ variables. Let $g_1(x,\alpha), \ldots, g_M(x,\alpha)$ be the set of polynomials occuring in $H$. The degree of $g_i$ is bounded by $s_n^{O(s_n)}$ and $M \leq s_n^{O(s_n(n+p))}$. Consider the $2^n M$ polynomials $p_{i,x}(\alpha)$ of the form $\alpha \mapsto g_i(x,\alpha)$, where $x \in \{0,1\}^n$ and $i \in \{1,\ldots,M\}$. The number of equivalences classes of $\equiv$ is bounded by the number of consistent sign conditions for the $p_{i,x}$. According to Theorem 11, this number is in turn bounded by

$$\left(2^n s_n^{O(s_n(n+p))} s_n^{O(s_n)}\right)^p.$$

This completes the proof of (4).

Let $B_{s,p}(n)$ be the set of boolean functions on $\{0,1\}^n$ which can be computed by an existential circuit of size $s$ with $p$ real constants. Since the number of circuis of size $S$ is singly exponential, It follows from (4) that

$$\#B_{s,p}(n) \leq 2^{O(p(n+p)s^2 \log s)}.$$

Now, set for instance $p(n) = n$ and $s(n) = n^{\log n}$. It is easily verified that $\#B_{s(n),p(n)}(n) < 2^{2^n}$ for some large enough $n_0$ and every $n > n_0$. Hence for $n > n_0$ there always exists a boolean function $L_n \subset \{0,1\}^n$ which is not in $B_{s(n),p(n)}(n)$. Consider the language $L = \cup_{n>n_0} L_n$. This language is not in $\mathrm{BP}(\mathrm{NP}_R)$ since for every $L' \in \mathrm{BP}(\mathrm{NP}_R)$, $L' \cap \{0,1\}^n$ is in $B_{s(n),p(n)}(n)$ for $n$ large enough. $\square$

This proof relies on the somewhat questionable possibility of using arbitrary real constants in the program of a real machine. In fact, Theorem 14 holds even without this possibility. The idea is simply to let real inputs play the role of real constants.

**Theorem 15** *Let $E \subset \mathbb{R}$ be an arbitrary set containing the constants 0 and 1. Then* $\mathrm{NP}_R(E)$ *is strictly included in* $\mathrm{EXP}_W(E)$.

*Proof.* There is essentially no difference between the inclusion $\mathrm{NP}_R(E) \subset \mathrm{EXP}_W(E)$ for an arbitrary $E$ and for the special case $E = \mathbb{R}$ considered in [12]. We give the proof here for the sake of completeness. In order to recognize a set $L \in \mathrm{NP}_R(E)$, construct fomula (3) in polynomial time. It follows from the results of Renegar [22] on the decision problem for the extended theory of the reals that given an input $x \in \mathbb{R}^n$, formula (3) can be decided in parallel polynomial time. Parallel polynomial time is included in weak exponential time since the number of processors is at most singly exponential and the degrees and coefficient size of all polynomials also have a singly exponential bound.

In order to see that the inclusion $\mathrm{NP}_R(E) \subset \mathrm{EXP}_W(E)$ is strict, we now show that $\mathrm{EXP}_W(0,1) \setminus \mathrm{NP}_R \neq \emptyset$. Consider a real language $\mathcal{L} \in \mathrm{EXP}_W(0,1)$ satisfying the following property: for every language $L \subset \{0,1\}^*$, there exists $y_L \in \mathbb{R}$ such that for every $x \in \{0,1\}^*$, $(x, y_L) \in \mathcal{L}$ iff $x \in L$ ($y_L$ thus plays the role of an arbitrary real constant). We claim that $\mathcal{L} \notin \mathrm{NP}_R$. Take indeed a language $L \subset \{0,1\}$ which is not in $\mathrm{BP}(\mathrm{NP}_R)$ (such a language exists according to Theorem 14). The restriction of $\mathcal{L}$ to inputs of the form $(x, y_L)$ clearly cannot be recognized in non-deterministic polynomial time. □

# 8 Application to artificial neural networks

## 8.1 Recurrent networks

We shall use the model of analog computation with neural nets proposed in [26]. Let us first recall briefly the main features of this model. The dynamics of a network of $n$ interconnected neurons $x_1, \ldots, x_n$ is defined by

$$x_i(t+1) = f_i\left(\sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{p} b_{ij} u_j(t) + c_i\right)$$

where $a_{ij}, b_{ij}, c_i \in \mathbb{R}$. The $u_j$'s are binary input lines: $u_j(t) \in \{0,1\}$. In the following we assume that there are only two input lines $u_1$ and $u_2$. These networks can be used to recognize boolean languages as follows. Starting from the initial state $x(0) = 0$, the input is read sequentially on $u_1$. $u_2$ is a validation line which remains active (in state 1) until the whole input has been read. The output is read on two designated output processors, say $x_1$ and $x_2$. $x_2$ is also a validation processor, active only when $x_1(t)$ is the actual result of the computation.

In order to apply the results of the previous sections, we assume that the output function $f_i$ of each processor is a (possibly discontinuous) piecewise-linear function.

**Theorem 16** *If a language $L \in \{0,1\}^*$ can be recognized in polynomial time by a recurrent neural network, then $L \in \mathrm{P/poly}$.*

*Proof.* This is a straightforward consequence of Theorem 4, since recurrent networks can be simulated with additions, comparisons, and multiplication by constants only. □

This theorem can be applied to discontinuous transition functions; this was not the case for the results of [26]. The authors of this paper showed that it is not possible to compute more than P/poly in polynomial time if the $f_i$'s are Lipschitz and polynomial-time approximable (this property holds for most continuous functions of practical interest).

## 8.2 Feedforward networks

It is shown in [20] that a family of acyclic circuits of constant depth and polynomial size using piecewise-polynomial activation functions (defined by real parameters) can be simulated by a family of threshold circuits which is still of constant depth and polynomial size. In this section, we consider families of networks of the same type, and show that they can be simulated for boolean inputs and outputs by families of threshold circuits with a "small" overhead in size and depth. Contrary to [20], we can obtain results for networks of more than constant depth ($\log n$ in the worst case and $n^{O(1)}$ in the best) using the additional assumption that the family is uniform, *i.e.*, it can be generated by a RTM in polynomial time in the weak BSS model.

**Definition 4** *A feedforward net is an acyclic graph of gates. Each gate $g$ computes an output of the form*

$$y = f_g(\sum_{i=1}^{k} w_i x_i - \theta) \tag{5}$$

*where $x_1, \ldots, x_k$ are the outputs of the predecessors of $g$. The output function $f_g$ is piecewise polynomial,* i.e., *there is a finite sequence $c_0 = -\infty < c_1 < \ldots < c_{l-1} < c_l = +\infty$ such that $f$ is polynomial on each interval $]c_i, c_{i+1}[$.*

*There are $n$ boolean input gates and a single gate of fan-out zero (the output gate). This gate is a threshold gate,* i.e., *its output function $f$ is such that $f(x) = 1$ if $x \geq 0$ and $f(x) = 0$ otherwise.*

**Definition 5** *A family of feedforward nets $C = (C_n)_{n\geq 1}$ (where $C_n$ has $n$ inputs) is* P-*uniform if*

1. *There exists $d \in \mathbb{N}$ such that for every $n$ and for every output function $f_g$ of $C_n$, the degrees of the polynomial pieces of $f_g$ are bounded by $d$.*

2. *There exists a RTM which on input $n$ outputs a complete description of $C_n$ in time polynomial in $n$ in the weak BSS model.*

The second condition implies in particular that $C_n$ is of polynomial size, and that each output function has a polynomial number of polynomial "pieces".

In a P-uniform circuit family, each parameter $W_i$ of a circuit $C_n$ is a rational fraction of the constants $\alpha_1, \dots, \alpha_p$ of the RTM which generates the family (these parameters are the weights and threshold of all gates, and the parameters of all output functions). By reducing these fractions to the same denominator, one can write

$$W_i = \delta P_i(\alpha_1, \dots, \alpha_p) \tag{6}$$

with $\delta = 1/Q(\alpha_1, \dots, \alpha_p)$ and $P_i, Q \in \mathbb{Z}[X_1, \dots, X_p]$. We are now in position to define a class of circuit families that can be efficiently simulated.

**Definition 6** *Let $C$ be a P-uniform family of feedforward nets generated by a RTM $\mathcal{M}$. For each $n$, each gate $g$ of $C_n$ and each input $(u_1, \dots, u_n) \in \{0,1\}^n$, the output $y_g$ of $g$ is a polynomial function $y_g = Q_g(\alpha_1, \dots, \alpha_p, \delta)$ of the parameters of $\mathcal{M}$ and the coefficient $\delta$ defined by (6). The family $C$ is said to be well-behaved if the $Q_g$'s have polynomial degrees and coefficients of polynomial size (in $n$).*

**Theorem 17** *A well-behaved family of feedforward nets of depth $D(n)$ can be simulated by a (non-uniform) family $(T_n)_{n\geq 1}$ of threshold circuits of polynomial size and depth $O(D(n))$. Moreover, the gates of $T_n$ have binary weights (from $\{-1, 1\}$).*

*Proof Sketch.* Just like in the proof of Theorem 4, we replace for each circuit $C_n$ the parameters $\alpha_1, \dots, \alpha_p, \delta$ by rational approximations $\overline{\alpha}_1, \dots, \overline{\alpha}_p, \overline{\delta}$ (these approximations must satisfy the constraint that for each input $u \in \{0,1\}^n$ and each gate $g$ of $C_n$, the polynomial piece of $f_g$ selected in the computation of $C_n$ on input $u$ is the same as with the actual parameters $\alpha_1, \dots, \alpha_p, \delta$). According to Definition 6 and theorem 6, these approximations have polynomial size. Hence it follows from Definition 6 that all the rational numbers arising in a computation of $C_n$ on any input $u \in \{0,1\}^n$ are of polynomial size.

We can thus conclude like in [20]: $C_n$ can be simulated by a threshold circuit $T_n$ of polynomial size and depth $O(D(n))$ since the basic arithmetic operations (comparison, multiplication and multiple addition of integers) can be performed by threshold circuits (with binary weights) of polynomial size and constant depth [6] (here we use the fact that the output functions have uniformly bounded degrees).

Note that in this construction we deal with equality cases with the technique of section 5 (hence in the complete construction another parameter $\overline{\epsilon}$ has to be used in addition to $\overline{\alpha_1}, \dots, \overline{\alpha_p}$ and $\overline{\delta}$). This avoids the problem of having to perform the algebraic computations of section 6 with shallow threshold circuits. □

We now give two examples of well-behaved circuit families.

**Corollary 3** *A P-uniform family $C = (C_n)_{n\geq 1}$ of feedforward nets of depth $O(\log n)$ can be simulated by a (non-uniform) family $(T_n)_{n\geq 1}$ of threshold circuits of polynomial size and depth $O(\log n)$. Moreover, the gates of $T_n$ have binary weights.*

*Proof Sketch.* We show that $C$ is well behaved. In (6), the polynomials $P_i$ have polynomial degree and coefficients of polynomial size. This is due to the fact that $C_n$ can be constructed in polynomial time in the weak BSS model, and that these polynomials are obtained by a reduction to the same denominator involving only a polynomial number of rational fractions.

Let $D$ be the maximum degree of the $P_i$'s, and $D_h$ the maximum degree of $Q_g$ where $g$ is a gate such that the longest path between an input unit and $g$ is at most $h$. According to (5), $D_{h+1} \leq D + d(D_h + D)$. Since $D = n^{O(1)}$, $d = O(1)$ and $h = O(\log n)$, it follows that $D_h = n^{O(1)}$. It is not difficult to show that the $Q_g$'s have also coefficients of polynomial size. Hence $C$ is well behaved and one can apply Theorem 17. □

**Corollary 4** *A* P*-uniform family of feedforward nets of depth $D(n)$ using piecewise-linear output functions can be simulated by a (non-uniform) family $(T_n)_{n\geq 1}$ of threshold circuits of polynomial size and depth $O(D(n))$. Moreover, the gates of $T_n$ have binary weights.*

*Proof Sketch.* We proceed like in the proof of Corollary 3. The induction relation $D_{h+1} \leq D + d(D_h + D)$ becomes $D_{h+1} \leq D_h + 2D$ for piecewise-linear output functions, thus $D_h = n^{O(1)}$ for $h = n^{O(1)}$. □

# 9 Final remarks

A perhaps unusual feature of the bound of Theorem 6 is that the number of inequations does not come into play. As was already mentioned in section 5, some explicit bounds use as parameter the total degree of the system (*i.e.*, the sum of the degrees of all the polynomials). In addition to Ronga's result, one can cite an upper bound on the diameter of a bounded semi-algebraic set [11] and a lower bound on the distance between two connected components of a semi-algebraic set [14]. It would be interesting to find out if the total degree can be replaced by the maximum degree in these two bounds.

On the other hand, these bounds are in a sense sharper than Theorem 6, since they exhibit a singly exponential dependence with respect to the dimension of the problem. It is not difficult to see that in Theorem 6 the dependence is in fact doubly exponential in $p$. It would also be interesting to determine whether it can be replaced by a singly exponential dependence.

Although $\mathrm{BP}(\mathrm{P}_W) = \mathrm{P/poly}$, it is not obvious at all whether $\mathrm{BP}(\mathrm{NP}_W) \subset \mathrm{NP/poly}$. Actually, we were only able to show in section 7 that not all boolean languages are in $\mathrm{BP}(\mathrm{NP}_W)$. The exact power of non-determinism can be determined for a weaker model in which multiplications and divisions are completely forbidden. It can be shown that the set of NP languages in this model is NP/poly [16]. The mathematics of this problem are much simpler since the relevant geometric objects are polyhedra instead of semi-algebraic sets. Similar results can also be shown for a real version of the Meyer-Stockmeyer polynomial time hierarchy [10].

## Acknowledgment

I am greatly indebted to Felipe Cucker for his help with the proof of Lemma 5.

## 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] R. Benedetti and J.-J. Risler. *Real algebraic and semi-algebraic sets.* Hermann, Paris, 1990.

[4] 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.

[5] J. Bochnak, M. Coste, and M.-F. Roy. *Géométrie algébrique réelle.* Springer-Verlag, 1987.

[6] J. Bruck and K.-Y. Siu. On the power of threshold circuits with small weights. *SIAM J. Disc. Math.*, 4(3):423–435, August 1991.

[7] M. Coste and M.-F. Roy. Thom's lemma, the coding of real algebraic numbers and the topology of semi-algebraic sets. *Journal of Symbolic Computation*, 5:121–129, 1988.

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

[9] F. Cucker, L. Gonzalez Vega, and F. Rosselló. On algorithms for real algebraic plane curves. In *Effective Methods in Algebraic Geometry*, volume 94 of *Progress in Mathematics*, pages 63–87. Birkhäuser, 1991.

[10] F. Cucker and P. Koiran. Computing over the reals with addition and order: Higher complexity classes. preprint.

[11] F. Cucker and F. Roselló. On the complexity of some problems for the Blum, Shub & Smale model. In *Proceedings of Latin'92*, volume 583 of *Lecture Notes in Computer Science*, pages 117–129. Springer-Verlag, 1992.

[12] F. Cucker, M. Shub, and S. Smale. Separations of complexity classes in Koiran's weak model. Technical Report RC 19346 (84217), IBM Research Division, 1994.

[13] John B. Goode. Accessible telephone directories. *Journal of Symbolic Logic*, 59(1), 1994.

[14] J. Heintz, T. Krick, M.-F. Roy, and P. Solerno. Single exponential time algorithms for basic constructions in elementary geometry. In *Proceedings of the* $10^{th}$ *Int. Conf. of the Chilean Comp. Sc. Soc.*, 1990.

[15] C. Kenyon and L. Lóvász. Algorithmic discrete mathematics. Technical Report CS-TR-251-90, Department of Computer Science, Princeton University, March 1990.

[16] 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.

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

[18] S. Lang. *Algebra*. Addison-Wesley, 1965.

[19] R. Loos. Computing in algebraic extensions. In Buchberger, Collins, and Loos, editors, *Computer Algebra, Symbolic and Algebraic Computation*, pages 173–187. Springer-Verlag, 1982.

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

[21] J. Milnor. On the Betti numbers of real varieties. *Proc. American Mathematical Society*, 15:275–280, 1964.

[22] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. parts I, II, III. *Journal of Symbolic Computation*, 13(3):255–352, March 1992.

[23] F. Ronga. Recherche de solutions d'inéquations polynomiales. In C. Hayat-Legrand and F. Sergeraert, editors, *Algorithmique, topologie et géométrie algébriques*, Astérisque 192, pages 11–16. Société Mathématique de France, 1990.

[24] A. Schönhage. On the power of random access machines. In *Proc.* $6^{th}$ *ICALP*, volume 71 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

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

[26] H. T. Siegelmann and E. D. Sontag. Neural networks with real weights: analog computational complexity. SYCON Report 92-05, Rutgers University, September 1992.

[27] H. T. Siegelmann and E. D. Sontag. Analog computation via neural networks. In *Proc.* $2^{nd}$ *Israeli Symposium on Theory of Computing and Systems*, 1993.

[28] S. Smale. Some remarks on the foundations of numerical analysis. *SIAM Review*, 32(2):211–220, June 1990.