On the power of real Turing machines over binary inputs

Felipe Cucker, Dima Grigoriev

On the power of real Turing machines over binary inputs

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

Dima Grigoriev†
Depts. of Comp. Science and Mathematics
Penn State University
University Park, PA 16802
USA
e-mail: dima@cse.psu.edu

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 [4] where a computational model —the real Turing machine— for dealing with the above problems was developed.

One research direction that has been studied intensively during the last two years is the computational power of real Turing machines over binary inputs. The general problem can be roughly stated in the following way. Let us consider a class C of real Turing machines that work under some resource bound (for instance polynomial time, branching only on equality, etc.). If we restrict these machines to work on binary inputs (i.e. finite words over $\{0,1\}$) they define a class of binary languages D. The question is, what can we say about D depending on C?

More formally, let us denote by $\mathbb{R}^\infty$ the direct sum of countably many copies of $\mathbb{R}$ and let $\mathcal{P}(\mathbb{R}^\infty)$ be the set of its subsets. Also, let us denote by Σ the subset $\{0,1\}$ of $\mathbb{R}$ and —as usual— by Σ^* the subset of $\mathbb{R}^\infty$ consisting

*Partially supported by DGICyT PB 920498, the ESPRIT BRA Program of the EC under contracts no. 7141 and 8556, projects ALCOM II and NeuroCOLT.
†Partially supported by Volkswagen–Stiftung.

of those vectors whose components are in Σ. Given any complexity class $\mathcal{C} \subseteq \mathcal{P}(\mathbb{R}^\infty)$, we define its Boolean part to be the class of binary languages

$$\mathrm{BP}(\mathcal{C}) = \{X \cap \Sigma^* : X \in \mathcal{C}\}$$

Our problem now can be stated as: given a complexity class of real sets $\mathcal{C}$ characterize $\mathrm{BP}(\mathcal{C})$.

A possible origin of the problem is the recent interest in the computational power of neural networks. The first results characterized the power of nets with rational weights working within polynomial time by showing that they compute exactly the sets in P (cf. [26]). The same problem was then considered for neural networks with real weights and it was shown that the power of these nets working within polynomial time is exactly P/*poly* (cf. [27],[28] and [21]).

This latter problem considers in a natural way a setting in which an algebraic model having real constants operates over binary inputs. A next step was then taken by P. Koiran who passed from a structured model — the neural net— to a general one —the real Turing machine. However, he did not deal with the real Turing machine as it was introduced in [4] but with a restricted version of it that can do only a moderate use of multiplication, namely all rational functions intermediately computed (in the input variables as well as in the machine's constants) must have degree and coefficient size bounded by the running time. For this weak model he considered the class $\mathrm{P_W}$ of sets accepted in polynomial time and he proved that $\mathrm{BP}(\mathrm{P_W}) = \mathrm{P}/poly$ (see [19]).

Subsequently, several papers exhibited new results on Boolean parts. In [12] it was shown that $\mathrm{BP}(\mathrm{PAR_W}) = \mathrm{PSPACE}/poly$ where $\mathrm{PAR_W}$ is the class of subsets of $\mathbb{R}^\infty$ decided in weak parallel polynomial time. Also, for additive machines (i.e. real Turing machines that do not perform multiplications at all), it was shown in [18] that $\mathrm{BP}(\mathrm{P_{add}}) = \mathrm{P}/poly$ and that $\mathrm{BP}(\mathrm{NP_{add}}) = \mathrm{NP}/poly$. Here $\mathrm{P_{add}}$ and $\mathrm{NP_{add}}$ denote the obvious classes but we recall that the nondeterministic guesses in this model are real numbers. Moreover, if the machines are order-free, i.e. they are required to branch only on equality tests, we now have that $\mathrm{BP}(\mathrm{P^{=}_{add}}) = \mathrm{P}$ and that $\mathrm{BP}(\mathrm{NP^{=}_{add}}) = \mathrm{NP}$ ([18]). These results were subsequently generalized in [10] to all the levels of the polynomial hierarchy constructed upon $\mathrm{NP_{add}}$ (or $\mathrm{NP^{=}_{add}}$).

None of the mentioned results was done for the (unrestricted) real Turing machine. In fact, for this case, it was even asked whether it existed a subset of Σ^* not belonging to the $\mathrm{BP}(\mathrm{P}_{\mathbb{R}})$ (cf. [13]). First steps in this direction

were done in [20] where it is shown that if we consider order-free machines then we have the inclusion $\mathrm{BP}(\mathrm{P}_{\mathbb{R}}^{=}) \subseteq \mathrm{BPP}$ (the class of sets decided by randomized machines in polynomial time with bounded probability error, see [1] ch. 6) as well as a positive answer to the question above. In fact if $\mathrm{PH}_{\mathbb{R}}$ is the polynomial hierarchy constructed upon $\mathrm{NP}_{\mathbb{R}}$, the existence of binary languages not belonging to $\mathrm{BP}(\mathrm{PH}_{\mathbb{R}})$ (and *a fortiori* nor to $\mathrm{BP}(\mathrm{P}_{\mathbb{R}})$) was also proved in [20].

The aim of this paper is to prove that $\mathrm{BP}(\mathrm{PAR}_{\mathbb{R}}) = \mathrm{PSPACE}/poly$ where $\mathrm{PAR}_{\mathbb{R}}$ is the class of sets computed in parallel polynomial time by (ordinary) real Turing machines. As a consequence we obtain the existence of binary sets that do not belong to the Boolean part of $\mathrm{PAR}_{\mathbb{R}}$ (an extension of the result in [20] since $\mathrm{PH}_{\mathbb{R}} \subseteq \mathrm{PAR}_{\mathbb{R}}$) and a separation of complexity classes in the real setting.

1 Some geometrical background

In the rest of the paper $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$ denote the sets of natural, integer, rational, real and complex numbers respectively. By $\mathbb{R}_{\mathrm{alg}}$ we denote the real closure of $\mathbb{Q}$, i.e. the field of all real algebraic numbers. Also, for any polynomial f with integer coefficients, we shall denote by $|\mathrm{coeff}(f)|$ the maximal absolute value of its coefficients.

The aim of this section is to show how to find real algebraic points in the connected components of non-empty open sets. We closelly follow [15]. Thus, let $g_1, \ldots, g_N \in \mathbb{Z}[X_1, \ldots, X_k]$ and let

$$V = \{x \in \mathbb{R}^k : g_1(x) > 0 \& \ldots \& g_N(x) > 0\}$$

be an open non-empty semialgebraic set. For the rest of this section we consider d a bound on the degree of each g_i and L be a bound for all $|\mathrm{coeff}(g_i)|$.

Lemma 1 ([15] Lemma 10) *Let $g = \prod_{i=1}^{N} g_i$. Let also d_1 be the degree of g and $L = |\mathrm{coeff}(g)|$. Then there exists a positive integer γ_1 such that any connected component of V has a non-empty intersection with the ball $B(R)$ where*

$$R = L^{{d_1}^{k\gamma_1}}$$

□

Let us recall now (see [5] section 9.5) that a point $a \in \mathbb{R}^k$ is a *critical point* for a function $f : \mathbb{R}^k \to \mathbb{R}$ when it satisfies

$$\frac{\partial f}{\partial X_1}(a) = \cdots = \frac{\partial f}{\partial X_k}(a) = 0$$

In this case the value $b = f(a)$ is said to be a *critical value* of f. In the case when f is a polynomial function Sard's lemma ([5] théorème 9.5.2 or [22]) implies that there are only a finite number of critical values of f.

This last fact was used in [15] (and in several subsequent papers) to reduce the dimension of non-empty semialgebraic sets to zero (avoiding thus cascading of projections) in the algorithm for deciding emptiness of semialgebraic sets.

Let us now consider the polynomials

$$g_0 = R^2 - \sum_{i=1}^{k} X_i^2$$

and

$$G = g_0 \prod_{i=1}^{N} g_i$$

We have that $\deg G = d_2 < Nd + 2$ and $L_2 = |\text{coeff}(G)| \leq L^{{d_2}^{O(k)}} (Ld^k)^{O(N)}$ due to lemma 1.

The following result gives a bound on the small critical values of G.

Lemma 2 *There exists a positive integer γ_2 such that for every non-zero critical value a of G we have $|a| > C^{-1}$ where*

$$C = L_2^{{d_2}^{k\gamma_2}}$$

Proof. Let us consider the system of equations in the variables $X_1, \ldots, X_k, Z$

$$G - Z = \frac{\partial G}{\partial X_1} = \cdots = \frac{\partial G}{\partial X_k} = 0$$

as well as its set of solutions $S \subseteq \mathbb{R}^{k+1}$. On any connected component of S the coordinate Z, being the critical value of G, is constant since G is continuous and due to Sard's lemma. Now, since the degrees and the coefficients of the polynomials appearing in this system are bounded by d_2 and $O(L_2 d_2)$ respectively, if we apply the quantifier elimination algorithm

given in [16] or [23] along $X_1, \ldots, X_k$ onto Z we get a finite set of points in $\mathbb{R}$ (just the critical values) such that each non zero one has absolute value greater than

$$\left(L_2^{{d_2}^{k\gamma_2}}\right)^{-1}$$

□

Remark 1 In the preceding proof the use of quantifier elimination is not strictly necessary. One can use instead the bounds for the representative points from the connected components of S given in the main theorem of [15].

Because of the preceding lemma we have that the algebraic set

$$W_0 = \{x \in \mathbb{R}^k : G(x) = C^{-1}\}$$

is a non-singular, closed, hypersurface with the property that each connected component of $V \cap B(R)$ contains at least one (bounded) connected component of W_0 (cf. [22]). Note that W_0 do not intersect the boundary of $B(R)$.

Now, lemma 5 of [15] asserts the existence of integers $0 \leq v_2, \ldots, v_n \leq (2d_2)^k$ such that the system

$$G - C^{-1} = \left(\frac{\partial G}{\partial X_2}\right)^2 - \frac{v_2}{(2d_2)^k k}\Delta = \cdots = \left(\frac{\partial G}{\partial X_k}\right)^2 - \frac{v_k}{(2d_2)^k k}\Delta = 0$$

where $\Delta = \sum_{i=1}^{k} (\frac{\partial G}{\partial X_i})^2$, has a finite number of solutions in $\mathbb{R}^k$. Moreover, each of these solutions is an absolutely irreducible 0–dimensional component of the variety in $\mathbb{C}^k$ given by this system of equations. Due to Bezout's inequality the number of real solutions is bounded by $(2d_2)^k$. Besides, (cf. lemma 4 in [15]), each bounded connected component of W_0 contains a point satisfying the system.

We can summarize the preceding results in the following theorem, that will be our main technical tool in the next section.

Theorem 1 *Let $g_1, \ldots, g_N \in \mathbb{Z}[X_1, \ldots, X_k]$ satisfy for every $i \leq N$ the bounds $\deg(g_i) \leq d$ and $|\text{coeff}(g_i)| \leq L$. Then, with the notations introduced above, there are integers $0 \leq v_2, \ldots, v_k \leq (2d_2)^k$ such that the set $W \subseteq \mathbb{R}^k$ defined by*

$$G - C^{-1} = \left(\frac{\partial G}{\partial X_2}\right)^2 - \frac{v_2}{(2d_2)^k k}\Delta = \cdots = \left(\frac{\partial G}{\partial X_k}\right)^2 - \frac{v_k}{(2d_2)^k k}\Delta = 0$$

is finite. Moreover, the number of its points does not exceed $(2d_2)^k$ *and every connected component of*

$$V = \{x \in \mathbb{R}^k : g_1(x) > 0 \,\&\, \ldots \,\&\, g_N(x) > 0\}$$

contains at least one point of W*.*

□

2 Computing with binary inputs

In this section we shall deal with real Turing machines (RTM for short) as they were introduced in [4]. The reader is referred to this paper for the definition and main properties of the above quoted computational model. We just recall that we denote $\mathbb{R}^\infty$ the direct sum $\bigoplus_{i\in\mathbb{N}} \mathbb{R}$ and by $\mathrm{P}_{\mathbb{R}}$ the class of subsets of $\mathbb{R}^\infty$ that can be decided by a RTM in polynomial time.

The goal of this section is to prove that the Boolean part of $\mathrm{P}_{\mathbb{R}}$ is included in PSPACE/*poly*, i.e. that any subset of $\Sigma*$ that can be decided in polynomial time by a real Turing machine can be decided by a (classical) Turing machine in polynomial space using a polynomial advice. In the next section we will prove a more general result namely, a similar inclusion for parallel real Turing machines. Because of clarity of exposition we will however first show the inclusion for the Boolean part of $\mathrm{P}_{\mathbb{R}}$.

We begin by recalling the definition of non-uniform classes as given in [17], that we extend to complexity classes over the reals.

Definition 1 *Let* $\mathcal{C} \subseteq \Sigma^*$ *(resp.* $\mathcal{C} \subseteq \mathbb{R}^\infty$*) be a class of sets and* $\mathcal{F}$ *be any class of functions from* $\mathbb{N}$ *to Sigma* (resp. from* $\mathbb{N}$ *to* $\mathbb{R}^\infty$*). The class* $\mathcal{C}/\mathcal{F}$ *is defined to be the class of all subsets* $B \subseteq \Sigma^*$ *(resp.* $B \subseteq \mathbb{R}^\infty$*) for which there exists a set* $A \in \mathcal{C}$ *and a function* $f \in \mathcal{F}$ *such that* $B = \{x : \langle x, f(|x|)\rangle \in A\}$*.*

We will be mainly interested in the case $\mathcal{F} = poly$, the class of functions f such that for some polynomial p we have $|f(n)| \leq p(n)$ for each $n \in \mathbb{N}$. For the boolean case, one can find the main properties and characterizations of classes like P/*poly* or PSPACE/*poly* (as well as of some other non uniform complexity classes) in chapter 5 of [1].

Theorem 2 *The inclusion* $\mathrm{BP}(\mathrm{P}_{\mathbb{R}}) \subseteq$ PSPACE/*poly holds.*

Proof. Let M be a RTM working in polynomial time, say n^q, and let $\alpha_1, \ldots, \alpha_k$ be its real constants.

For any $n \in \mathbb{N}$ the machine M has an associated algebraic computation tree $T_{M,n}$ having depth n^q and size bounded by 2^{n^q}. To each branching node i of this tree corresponds a rational function $g_i \in \mathbb{Q}(\alpha_1, \ldots, \alpha_k)(X_1, \ldots, X_n)$ such that the branching is done according to whether the actual input $x \in \mathbb{R}^n$ satisfies $g_i(x) \geq 0$ or $g_i(x) < 0$.

The idea of the proof is to find $\beta_1, \ldots, \beta_p \in \mathbb{R}_{\text{alg}}$ such that, for every $x \in \Sigma^n$ the path followed by x in the tree $\widetilde{T}_{M,n}$ obtained by replacing the constants α_j by the β_j is the same as the one followed in $T_{M,n}$. This ensures that the tree $\widetilde{T}_{M,n}$ accepts the same subset of Σ^n than $T_{M,n}$. On the other hand, we will require some codification of the β's that allows us to perform the operations in $\widetilde{T}_{M,n}$ in PSPACE together with a short way of writing this codification —that make possible to give it as a polynomial advice.

Before obtaining a description of the β's let us do a last modification on $T_{M,n}$ that was first used in [19]. Let I be the set of branching nodes of $T_{M,n}$. For every $i \in I$ and every $x \in \Sigma^n$ we consider the rational functions

$$g_{i,x} \in \mathbb{Q}(Z_1, \ldots, Z_k)$$

obtained by replacing the real constants $\alpha_1, \ldots, \alpha_k$ by indeterminates $Z_1, \ldots, Z_k$ and the indeterminates $X_1, \ldots, X_n$ by the binary values $x_1, \ldots, x_n$. We obviously have for every $x \in \Sigma^n$ that $g_i(x) = g_{i,x}(\alpha_1, \ldots, \alpha_k)$. Let $\tau_{i,x}$ be these values. The accepted subset of Σ^n can be characterized by the set of signs

$$\sigma_{i,x} = \text{sign}(\tau_{i,x})$$

where $\text{sign}(z) = 1$ if $z \geq 0$ and 0 otherwise. Now, for some i, x the element $\tau_{i,x}$ can be zero. However, since the set of values

$$\{\tau_{i,x} : i \in I, x \in \Sigma^n\}$$

is finite, there exist an $\varepsilon > 0$ such that all the negative values in the above set are strictly smaller than $-\varepsilon$ and for this ε the following equivalences hold

$$\begin{aligned} g_{i,x}(\alpha_1, \ldots, \alpha_k) \geq 0 \quad &\text{iff} \quad g_{i,x}(\alpha_1, \ldots, \alpha_k) + \varepsilon > 0 \\ g_{i,x}(\alpha_1, \ldots, \alpha_k) < 0 \quad &\text{iff} \quad g_{i,x}(\alpha_1, \ldots, \alpha_k) + \varepsilon < 0 \end{aligned}$$

Thus, replacing the tests $g_i(X) \geq 0$ by $g_i(X) + \varepsilon \geq 0$ we have that the new computation tree (having real constants $\alpha_1, \ldots, \alpha_k, \varepsilon$) satisfies the following property: for every $x \in \Sigma^n$ all the test values are different from zero.

Assuming that the rational functions $g_{i,x}(Z_1, \ldots, Z_k)$ are polynomials (something that we can do by just replacing it by the product of its numerator and denominator), we can resume the above remarks in the following way: the elements $\alpha_1, \ldots, \alpha_k, \varepsilon$ satisfy a system of polynomial inequations of the form

$$g_{i,x} > 0 : i \in I, x \in \Sigma^n \tag{1}$$

(we changed the sign of some $g_{i,x}$ in order to have all the inequalities in the same form) and any other real numbers $\beta_1, \ldots, \beta_k, \xi$ satisfying this system will produce, when used as constants in the tree $T_{M,n}$, the same outcome for every $x \in \Sigma^n$.

We can now describe how to obtain such numbers.

First we construct the g_0 and the G of the preceding section for the set of polynomials $\{g_{i,x} \in \mathbb{Z}[Z_1, \ldots, Z_k, Y] : i \in I, x \in \Sigma^n\}$. Then, applying theorem 1 we deduce the existence of integer vectors $\vec{v} = (v_2, \ldots, v_k, v_{k+1})$ such that the set W described there is finite and non empty. Let $\vec{v^*}$ be the first such vector for the lexicographical ordering in $\mathbb{N}^k$ and let W^* be its corresponding set of solutions. We then have that any connected component of the semialgebraic set V given by the system (1) contains a point of W^*. Thus, we take $\beta_1, \ldots, \beta_k, \xi$ be any point of W^* belonging to V, and we distinguish it among the other points of W^* by its position p for the lexicographical ordering in $\mathbb{R}^{k+1}$. Note that, from the equations defining W^* and this p we can code (cf. [16] or [23]) each coordinate of this point (see complexity analysis below).

The following non-uniform parallel algorithm decides then the same language as M when restricted to binary inputs.

input$(a_1, \ldots, a_n)$
get the advice p corresponding to n
for all $x \in \Sigma^n$ **in parallel do** (s1)
  for all path γ **in parallel do**
    for all i branch node in γ **do**
      compute the polynomial $g_{i,x}$
    od
  od
od
compute g_0 and G (s2)

compute C (s3)
compute $\vec{v^*}$ (s4)
code the coordinates of the p^{th} point β of (s5)
the set W^* given by $\vec{v}, C$ and G
simulate the computation of M over $a_1, \ldots, a_n$ (s6)
replacing the $\alpha_1, \ldots \alpha_k, \varepsilon$ by the point β coded in (s5)

Let us estimate the complexity of the algorithm above. As we have seen, the number of nodes of the tree $T_{M,n}$ is bounded by 2^{n^q}. Therefore, the number of polynomials $g_{i,x}$ is bounded by the same quantity. Each of these polynomials is computed by a straight-line program of length n^q and thus, we get again a bound of 2^{n^q} for their degrees and of $2^{2^{n^q}}$ for the absolute value of their coefficients. The degree d_2 of G is then bounded by $2^{n^q}.2^{n^q} = 2^{O(1)n^q}$ and the absolute value of its coefficients L_2 by

$$(2^{2^{n^q}})^{2^{n^q}} = 2^{2^{2n^q}}$$

(and thus, by $2^{n^{2q}}$ in bit length). We can then —according to theorem 1— bound by

$$(2d_2)^{k+1} = 2^{O(1)n^q}$$

the integers $v_2, \ldots, v_k, v_{k+1}$ and by

$$(2d_2)^{k+1} = 2^{O(1)n^q}$$

the number of points in W^*. A first consequence of these two last upper bounds is the fact that the advice above has polynomial size.

Concerning the running time, it is clear that step (s1) can be done in polynomial time using an exponential number of processors because, given an $x \in \Sigma^n$ and a path γ the —at most— n^q polynomials that appear in that path have exponential degree in a constant number ($k+1$ in fact) of variables and therefore, an exponential number of monomials. Any arithmetical operation between two such polynomials can be done within these resources and we have a polynomial number of such operations.

The product G is computed with a binary tree of products having polynomial depth. Since each product can be done in parallel polynomial time the same applies for the whole tree and then for step (s2). A similar remark holds for the constant C and thus for step (s3).

The determination of $\vec{v^*}$ can be done in parallel polynomial time since we check for all possible vectors $\vec{V}$ whether the dimension of the resulting

W is zero, and then we select the first $\vec{v}$ that gives a positive answer. Note that the determination of the dimension of each W can be done in PSPACE just combining the main idea of [14] section 6 with the parallel algorithms given in [16] or [23].

For step (s5) one can apply the algorithms given in [16] or [24]. However, we remark here that a cylindrical algebraic decomposition together with the coding *à la Thom* (see [7] for the algorithms, and [25], [11] for complexity analysis) suffices because the double exponential behaviour of this algorithm is only in the number of variables —which is constant in our case— being NC in the rest of the parameters. This results on a procedure for (s5) working in parallel polynomial time.

Finally, note that each arithmetical operation of M is translated in step (s6) into an operation of elements in $\mathbb{Z}[Z_1, \ldots, Z_k]$ and it is done then also in parallel polynomial time. On the other hand, at each test of the form $g(Z_1, \ldots, Z_k) \geq 0$ we use the same algorithm of step (s5) for determining the sign of $g(Z_1, \ldots, Z_k) + Y$ on the point coded in (s5).

The above considerations show that the described algorithm runs in parallel polynomial time. Since this is equivalent to polynomial space ([6], [2] ch.4) we have shown that the set decided by the algorithm above belongs to PSPACE/*poly*. □

3 Binary inputs for parallel real Turing machines

Our next goal is to extend our previous result to the class $\mathrm{PAR}_{\mathbb{R}}$ of sets decided in parallel polynomial time. We recall from [9] the definition of a computational model for parallelism in the real Turing machine setting together with the complexity class it defines when restricted to polynomial time.

Definition 2 *An algebraic 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 one node with outdegree 0 called output node. 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 gate 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.

Definition 3 *For an algebraic circuit $\mathcal{C}$, we define its size to be the number of gates in $\mathcal{C}$ and its depth to be the length of the longest path from some input or constant gate to the output gate.*

Definition 4 *Given an algebraic 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 one the output gate.*

Definition 5 *Let $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of algebraic circuits. We shall say that the family is P–uniform if there exists a real Turing machine M that generates the encoding of the i^{th} gate of $\mathcal{C}_n$ with input (i, n) in time polynomial in n.*

Definition 6 *We shall say that a set S can be decided in parallel polynomial time ($S \in \mathrm{PAR}_{\mathbb{R}}$ 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.*

Remark 2 It is possible to define ([9]) parallel polynomial time in a different way, namely, by putting an exponential number of RTM to work together in polynomial time. One can prove however, that this model defines the same class $\mathrm{PAR}_{\mathbb{R}}$ we just introduced.

Before going into the next theorem we will recall a result concerning the number of satisfiable sign conditions of a polynomial system.

Lemma 3 (Lemma 1 of [14]) *Let $f_1, \ldots, f_s \in \mathbb{R}[X_1, \ldots, X_k]$ be a finite family of polynomials and $D = \sum_{i=1}^{s} degree(f_i)$. Then the number of satisfiable systems of the form*

$$f_1(X_1, \ldots, X_k)\sigma_1 \ \& \ldots \& \ f_s(X_1, \ldots, X_k)\sigma_s$$

where σ_i belongs to $\{\geq 0, > 0\}$ for $i = 1, \ldots, s$ is bounded by $D^{O(k)}$.

□

Theorem 3 *The equality* $\mathrm{BP}(\mathrm{PAR}_{\mathbb{R}}) = \mathrm{PSPACE}/poly$ *holds.*

Proof. Let S be a set in $\mathrm{PAR}_{\mathbb{R}}$ and $\{\mathcal{C}_n\}$ be the family of circuits deciding S. Let also M be the RTM that generates these circuits and $\alpha_1, \ldots, \alpha_k$ be its real constants.

Given any $n \in \mathbb{N}$ we consider for any sign gate i of $\mathcal{C}_n$ and any binary string $x \in \Sigma^n$ the rational function $g_{i,x,\alpha} \in \mathbb{Q}(\alpha_1, \ldots, \alpha_k)(X_1, \ldots, X_n)$ that the gate receives as input. Note that the coefficients of $g_{i,x,\alpha}$ depend on x and α since they depend on the output of previous sign gates. Since the number of possible answers to previous sign gates is doubly exponential we obtain a doubly exponential number of rational functions and therefore we can not apply directly the construction of theorem 2. However, we can use lemma 3 to reduce this number.

Let us fix $x \in \Sigma^n$ and plug x in the input gates of $\mathcal{C}_n$. This will make us to consider rational functions in $\mathbb{Q}(Z_1, \ldots, Z_k)$. Let also n^q be a bound on the depth of $\mathcal{C}_n$. At depth 1, there are at most 2^{n^q} sign gates whose input functions have degree bounded by 1. By lemma 3, the number of possible outputs of these sign gates is bounded by

$$(2^{n^q})^{O(k)} = 2^{O(k)n^q}$$

For each set of outputs ω at depth 1, we consider the sign gates at depth 2. There are at most 2^{n^q-1} of them and their associated functions have degree bounded by 2. Thus, again by lemma 3, we bound by

$$(2^{n^q})^{O(k)} = 2^{O(k)n^q}$$

the number of possible outputs for ω. Multiplying both expresions we deduce that the total number of possible outputs at depths 1 and 2 is bounded by

$$2^{2O(k)n^q}$$

Inductively, we prove that the number of possible outputs over all the sign gates is bounded by

$$2^{O(k)n^q n^q} = 2^{O(k)n^{2q}}$$

a number which is singly exponential in n.

Let us then consider for any $x \in \Sigma^n$ the set of all rational functions $g_{i,x,\omega}(Z_1, \ldots, Z_k)$ obtained by varying i over all sign gates of $\mathcal{C}_n$ and ω over all possible outputs of the set of sign gates. If we now consider this set for

any $x \in \Sigma^n$ we will have that the set decided by the circuit $\mathcal{C}_n$ is determined by the signs that the functions in this set take when evaluated at $\alpha_1, \ldots, \alpha_k$.

As in theorem 2, we can assume the functions $g_{i,x,\omega}$ to be polynomials and we can also assume that they do not vanish on $\alpha_1, \ldots, \alpha_k$ by adding a new real number ε.

Since the number of polynomials $g_{i,x,\omega}$ is singly exponential in n we can apply the same method of theorem 2. Note however that the corresponding step (s1) will now be required to select for any $x \in \Sigma^n$ and any depth l the possible sign conditions for the test gates at depth l. This is done sequentially in l in order to avoid dealing with a doubly exponential number of sign conditions. Once these possible sign consitions are known, the rest of the algorithm works like the one in theorem 2 simulating the circuit $\mathcal{C}_n$ instead of the tree. This shows that the binary elements of S are a language in PSPACE/*poly*.

On the other hand, the inclusion of PSPACE/*poly* in $\mathrm{BP}(\mathrm{PAR}_{\mathbb{R}})$ is trivial. □

An immediate corollary of the theorem above is the following separation left open in [12]. Recall that $\mathrm{EXP}_{\mathrm{W}}$ is the class of subsets of $\mathbb{R}^\infty$ accepted by RTM in weak exponential time, i.e. in exponential time but such that for all intermediately computed rational function g $\deg(g)$ and the bit length of $|\mathrm{coeff}(g)|$ are exponentialy bounded (see [19] or [12] for a formal definition of the weak model).

Corollary 1 *The inclusion* $\mathrm{PAR}_{\mathbb{R}} \subset \mathrm{EXP}_{\mathrm{W}}$ *is strict.*

Proof. The Boolean part of $\mathrm{EXP}_{\mathrm{W}}$ is the class of all subsets of Σ^*. Therefore, it strictly contains PSPACE/*poly*. □

Remark 3 The corollary above improves the separation $\mathrm{PAR}_{\mathbb{R}} \neq \mathrm{EXP}_{\mathbb{R}}$ shown in [8]. In this latter case, the fact that a real Turing machine working in exponential time can produce polynomials of doubly exponential degree (while a circuit of polynomial depth can not) togheter with an irreducibility argument sufficed to show the separation. The arguments used now are much more delicate and, somehow surprisingly, pass throught the boolean part of these classes.

One can still improve a bit theorem 3 by allowing the real machine to take advice.

Theorem 4 *The equality* $\mathrm{BP}(\mathrm{PAR}_{\mathbb{R}}/poly) = \mathrm{PSPACE}/poly$ *holds.*

Proof. The polynomial advice in $\mathrm{PAR}_{\mathbb{R}}/poly$ introduces a polynomial number of real constants, let's say n^h, for each input size n. One can now simply check that replacing the constant value k in the proof of theorem 3 by n^h does not afect the exponential character of the bounds there and thus, the same arguments apply. The only limitation is that in steps (s5) and (s6) one can not use cylindrical algebraic decomposition (because of the exponential dependence it has in the number of variables for its parallel running time) and it is restricted to use the "faster" algorithms given in [16] and [23]. □

Remark 4 Theorems 3 and 4 are rather surprising since they show that multiplication or non-uniformity (under the form of a polynomial advice function) do not help in the presence of parallelism to decide binary sets. Note that results weaker than theorem 3 namely, that the Boolean part of $\mathrm{PAR}_{\mathrm{add}}$ (where no multiplications are allowed) or of $\mathrm{PAR}_{\mathrm{W}}$ (where few of them are allowed) coincide both with $\mathrm{PSPACE}/poly$ were proved in [10] and [12].

On the other hand a main question that remains open is whether $\mathrm{BP}(\mathrm{P}_{\mathbb{R}}) = \mathrm{PSPACE}/poly$. We know that this Boolean part contains $\mathrm{P}/poly$ but its exact power is still to be determined. Note that for the integer RAM's it is known that the computational power of this model in polynomial time is exactly PSPACE for several sets of primitive operations. However, in all these cases, there is a primitive operation that can not be efficiently simulated by a real Turing machine. Thus, for instance, it is shown in [3] that integer RAM's with operations $(+, -, *, \div)$ have the power of PSPACE. However, the simulation of the integer division by a real Turing machine over integers of exponential length take exponential time and therefore the arguments of [3] can not be used to show the inclusion $\mathrm{PSPACE}/poly \subseteq \mathrm{BP}(\mathrm{P}_{\mathbb{R}})$.

Ackowledgement Thanks are due to Pascal Koiran for pointing to us the possibility of allowing advice in the real complexity classes that lead from theorem 3 to theoerem 4.

References

[1] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, 11. Springer-Verlag, 1988.

[2] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs on Theoretical Computer Science, 22. Springer-Verlag, 1990.

[3] A. Bertoni, G. Mauri, and N. Sabadini. Simulations among classes of random access machines and equivalence among numbers succintly represented. *Annals of Disc. Math.*, 25:65–90, 1985.

[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] A. Borodin. On relating time and space to size and depth. *SIAM J. on Computing*, 6:733–744, 1977.

[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. On the complexity of quantifier elimination: the structural approach. *The Computer Journal*, 36(5):400–408, 1993.

[10] F. Cucker and P. Koiran. Computing over the reals with addition and order: higher complexity classes. Technical Report 94-8, DIMACS, 1994.

[11] F. Cucker, H. Lanneau, B. Mishra, P. Pedersen, and Roy. M.-F. NC algorithms for real algebraic numbers. *AAECC*, 3:79–98, 1992.

[12] F. Cucker, M. Shub, and S. Smale. Complexity separations in Koiran's weak model. *Theor. Comp. Sc.* To appear.

[13] J.B. Goode. Accessible telephone directories. *Journal of Symb. Logic*, 59(1):92–105, 1994.

[14] D.Yu. Grigoriev. Complexity of deciding Tarski algebra. *J. Symb. Comput.*, 5:65–108, 1988.

[15] D.Yu. Grigoriev and N.N. Vorobjov. Solving systems of polynomial inequalities in subexponential time. *J. Symb. Comput.*, 5:37–64, 1988.

[16] J. Heintz, M.-F. Roy, and P. Solerno. Sur la complexité du principe de Tarski-Seidenberg. *Bulletin de la Société Mathématique de France*, 118:101–126, 1990.

[17] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982.

[18] P. Koiran. Computing over the reals with addition and order. *Theor. Comp. Sc.* To appear.

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

[20] P. Koiran. A weak version of the Blum, Shub & Smale model. Technical Report 94-10, DIMACS, 1994.

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

[22] J. Milnor. *Topology from the differentiable viewpoint.* University Press of Virginia, 1965.

[23] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part III. *Journal of Symbolic Computation*, 13(3):329–352, March 1992.

[24] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I. *Journal of Symbolic Computation*, 13(3):255–299, March 1992.

[25] M.-F. Roy and A. Szpirglas. Complexity of computation on real algebraic numbers. *Journal of Symbolic Computation*, 7:39–51, 1990.

[26] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. In *Proc. Fifth ACM Workshop on Computational Learning Theory*, July 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] H. T. Siegelmann and E. D. Sontag. Neural networks with real weights: Analog computational complexity. *Theoretical Computer Science*, 1993. to appear.