Laboratoire de l'Informatique du Parallélisme

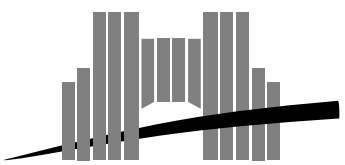
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

On NC-real complexity classes for additive circuits and their relations with NC

Michel Cosnard & Martín Matamala

Octobre 1993

Research Report N^{o} 93-35



Ecole Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07, France,
Téléphone : + 33 72 72 80 00; Télécopieur : + 33 72 72 80 80;
Adresses électroniques :
lip@frensl61.bitnet; lip@lip.ens−lyon.fr (uucp).

On NC-real complexity classes for additive circuits and their relations with NC

Michel Cosnard & Martín Matamala

Octobre 1993

Abstract

Based on the results of Blum, Shub and Smale [1], Meer [5], Cucker and Matamala [3] and Koiran [7], we develop the study of real computation models restricted to additive operations. More specifically we introduce some complexity classes defined by algebraic circuits and we study their relationships with the real computation model. We show that the languages accepted by nonuniform additive circuits of polynomial size and polylogarithmic depth are those accepted by uniform additive circuits of polynomial size and polylogarithmic depth with advice. Moreover, we prove that binary languages accepted by real uniform circuits of polynomial size and polylogarithmic depth, when the test nodes in the circuit are equality test, are the languages belonging to NC; when the test nodes are inequality test, the class obtained is NC/Poly. We also prove that the class defined by family of algebraic circuits with polynomial size and polylogarithmic depth is strictly contained in the class defined by real additive Turing machines working in polynomial time.

Keywords: Parallel computation complexity, real computational models, algebraic circuits.

Résumé

À partir des résultats de Blum, Shub et Smale[1], Meer [5], Cucker et Matamala [3] et Koiran[7], nous présentons une étude des modèles de calculs réels restreints aux opérateurs additifs. Nous définissons des classes de complexité à l'aide des circuits algébriques et nous étudions leurs relations avec le modèle réel. Nous montrons que les langages acceptés par des circuits additifs non uniformes de taille polynomiale et de profondeur polylogarithmique sont les mêmes que ceux acceptés par des circuits additifs uniformes avec conseil de taille polynomiale et de profondeur polylogarithmique. De plus, nous démontrons que la classe des langages binaires acceptés par des circuits réels uniformes de taille polynomiale et de profondeur polylogarithmique est la classe NC si les noeuds de test sont des tests d'égalité et la classe NC/Poly si les tests d'inégalité sont autorisés. Nous montrons enfin que la classe définie par les familles de circuits algébriques de taille polynomiale et de profondeur polylogarithmique est strictement incluse dans la classe des langages acceptés par une machine de Turing additive reélle en temps polynomial.

Mots-clés: Complexité des calculs parallèles, modèles de calculs réels, circuits algébriques.

On NC-real complexity classes for additive circuits and their relations with NC*

Michel Cosnard & Martín Matamala
Laboratoire de l'Informatique du Parallélisme
Ecole Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07, France.

October 27, 1993

Abstract

Based on the results of Blum, Shub and Smale[1], Meer[5], Cucker and Matamala[3] and Koiran[7], we develop the study of real computation models restricted to additive operations. More specifically we introduce some complexity classes defined by algebraic circuits and we study their relationships with the real computation model. We show that the languages accepted by nonuniform additive circuits of polynomial size and polylogarithmic depth are those accepted by uniform additive circuits of polynomial size and polylogarithmic depth with advice. Moreover, we prove that binary languages accepted by real uniform circuits of polynomial size and polylogarithmic depth, when the test nodes in the circuit are equality test, are the languages belonging to NC; when the test nodes are inequality test, the class obtained is NC/Poly. We also prove that the class defined by family of algebraic circuits with polynomial size and polylogarithmic depth is

* Supported by the Programme de Recherche Coordonnée C3 and by Grant Marie Curie ERBCISTGT 920031

†Adress: Laboratoire de l'Informatique du Parallélisme Ecole Normale Supérieure de Lyon 46, Allée d'Italie, 69364 Lyon Cedex 07, France. From December 93: Martín Matamala, Departamento de Ingeniería Matemática, U. de Chile, Casilla 170-correo 3 Santiago, Chile

strictly contained in the class defined by real additive Turing machines working in polynomial time.

1 Introduction

The real computation model was introduced in [1] where the existence of universal machines and complete problems when the machine can guess a sequence of real numbers was proved. Later, many authors have studied restrictions to this model [4, 5, 3]. In [5], Klaus Meer proved that when only linear operations are considered then the class NP and P can be separated. Moreover, Cucker and Matamala [3] have studied this model when the machine can guess a sequence of binary numbers and proved that the alternation of binary quantifiers is strictly less powerful than the alternation of real quantifiers. The existence of complete problems for this kind of nondeterminism is also proved when the operations in the machine are only the additions and subtractions. The models are called additive machines and the classes NP_{add} and P_{add} are defined in the usual way. Here, we introduce the class NC_{add} of the languages recognized by an uniform family of additive circuits of polylogarithmic depth and polynomial size. Our principal results are that the classes NC_{add} are strictly contained in the classes P_{add}. In section 2 we introduce the definitions and prove one result on circuit complexity. Later, in section 3, we study circuits working on binary input. When the circuits have test gates $x = 0?$ the complexity class obtained is NC, when the test circuits are $x \geq 0?$ the complexity class is NC/Poly. In section 4 we prove the strict inclusions of NC_{add} into P_{add}.

2 Definitions

We recall from [1, 3, 7] that an additive real Turing machine consists of an input space $I = R^{\infty}$(which is defined by $R^{\infty} = \oplus R^n$), an output space $O = R^k$, $(k \leq \infty)$ and a state space $S = I\!N \times I\!N \times R^{\infty}$, together with a connected directed graph whose nodes labeled $1 \ldots N$ correspond to the set of different instructions of the machine and are of one of the five following types: input, output, computation, test and move nodes. There is also a set of real constant $a_1 = 1, a_2, ..., a_r$ which can be used by a computation node. We describe nodes behavior in more details.

Input nodes. There is only one input node labeled 1. Associated with this node there is a next node $\beta(1)$, and the input map $g_I : I \to S$.

Output nodes. There is only one output node labeled 2. It has no next nodes. Once it is reached, the computation halts, and the output map $g_O : S \to O$ places the result of the computation in the output space.

Computation nodes. Associated with a node m of this type there is a next node $\beta(m)$ and a map $g_m : S \to S$. The g_m is of the form $g_m(i,j,z) = (i'(i,j), j'(i,j), z'(z))$, with $i'(i,j) = i+1$ or 1, $j'(i,j) = j+1$ or 1, and z' is an additive combination on a finite number of coordinates and a certain number of constants, and the identity on the others.

Test nodes. There are two nodes associated with a node m of this type: $\beta^1(m)$ and $\beta^2(m)$. Depending on the kind of test nodes we define two different models: equality model and inequality model. In the former the next node is $\beta^1(m)$ if $z_1 \neq 0$ and $\beta^2(m)$ otherwise. In the latter, the next node is $\beta^1(m)$ if $z_1 \geq 0$ and $\beta^2(m)$ otherwise. When no restriction is made in the node numbers the equality model is a particular case of inequality model.

Move nodes or fifth nodes. Nodes m of this type have a unique next node $\beta(m)$. If the current element of S is $(i,j,z_1,\ldots)$ it operates replacing z_j by z_i in the j-th place of the vector R^∞ in S.

We shall take as input map g_I the one placing the input in the coordinates z_s with s odd (thus we reserve the even coordinates to leave work space), and its size in z_2. All the other real coordinates of S are initially set to 0. Also, the input map sets the initial values of i and j in S to be 1. When $O = R^\infty$ we will take g_O as the identity map on the real coordinates of S. In the case $O = R^k$ with finite k we shall take g_O as the identity restricted to the first k real coordinates of S. For more details about the model the reader should consult [1].

We give an elementary result showing the computation capabilities of an additive machine.

For an element $x \in R^\infty$, we define $|x|$ as the largest coordinate which is non-null and we call $|x|$ the size of x. We denote $x \in R^\infty$ of size n by $x = x_1, ..., x_n$ where x_n is the largest coordinate non-zero. On inputs of size n we call $s(n)$ the largest non-zero coordinate and $t(n)$ the maximum running time of the machine. We say that a machine works in polynomial time if $t(n)$ is a polynomial. The class P_{add} is the set of all subsets in R^∞ that can be accepted by an additive machine working in polynomial time. We say that

it works in logarithmic space if $s(n)$ is $O(log(n))$.

Lemma 1 *Let M be an additive machine with constants $a_1 = 1, a_2, ..., a_r$ and let z_{kt} be the value of the coordinate k at time t of the state vector when M works on the input $x = x_1, ..., x_n$. Then z_{kt} can be decomposed in:*

$$z_{kt} = \sum_{j=1}^{r} \alpha_{ktj} a_j + \sum_{j=1}^{n} \beta_{ktj} x_j$$

where $\alpha_{ktj}, \beta_{ktj}$ are bounded by 2^t.

Proof This result was proved in [4] and later in [3]. The proof proceeds by induction on t. ■

This lemma says that only additive combinations of the constants and inputs can be obtained as outputs. In spite of the last result an additive machine with inequality test has an infinite memory capacity. In fact, the next result proves that an additive machine may interpret any binary codification.

Lemma 2 *Let α be given by*

$$\alpha = \sum_{j=1}^{\infty} 2^{-j} \alpha_j$$

where α_j belong to $\{0, 1\}$. Then there exists M an additive machine with inequality tests having 1 as constant so that when $\alpha...\alpha = \alpha^n$ is taken as input, $\alpha_1, ..., \alpha_n$ is obtained as output in time $O(n)$ and space $O(1)$.

Proof We make the following steps for an additive machine with tests $x \geq 0$?.

```
1     α' ← α
2     For i = 1 to n
3         α' ← 2α'
4         if α' ≥ 1 then print 1, α' ← α' − 1
5         else print 0
```

Clearly at step t, α' begins by α_t and it corresponds to the suitable branching in line 4. ■

Observe that we may encode the discrete structure of any machine as a real number(i.e., the description of the graph giving the instructions) and then a real vector may describe completely a machine. So, the machine given in lemma 2 is an universal decoder. Now, we define the complexity class for additive circuits which are defined as follows.

An *additive circuit* C is an acyclic graph whose nodes are labeled by operations. These operations are of five types: input operations, constant operations, arithmetic operations, test operations and output operations. The description of a node operation is as follows:

Input nodes receive from the exterior an initial real value. They have only an output edge and no input edges.

Constant nodes are defined by a real constant(which is fixed during the computation). They have only an output edge and not input edges. We suppose that all the circuits have at least one constant node with value 1.

Arithmetic nodes can be addition or subtraction. They receive two inputs from nodes i_1 and i_2 and give as output $x * y$ where x(resp. y) is the value of the i_1(resp. i_2)th node with $* = +$(resp. $* = -$) if the node is an addition(resp. subtraction)node.

Test nodes receive a real input x and make the test $x \geq 0$? (or the test $x = 0$?) giving 1 in the affirmative case and 0 otherwise. Like in additive machines we make the difference between equality circuits(those with $x = 0$? test) and inequality circuits(those with $x \geq 0$? tests).

Output node is the answer of the circuit. It has no output edges and one input edge. The value of the node connected to it is taken as value for this node. We suppose that the previous node is always a test node.

The depth of a node is the maximum depth of its inputs, plus one. We consider that input nodes and constant nodes have depth 0. The depth of the output node is called the circuit depth. The size of the circuit is the total number of nodes.

The circuit works over an input $u = u_1...u_n$ by setting the input nodes at values u_i and by computing the value at depth t using the values computed at depth $t-1$ in a recursive fashion. We say that C accepts u if the output node takes the value 1. We describe this fact by writing $C(a, u) = 1$ which should be read as "the circuit C with a constant vector a on input u answers

1".

A description of a circuit is given by:

$$\phi(C) = (g_1\#t_1\#g_{11}\#g_{12})\#...\#(g_L\#t_L\#g_{L1}\#g_{L2})$$

where g_i is a node number, t_i is its type ($0 \rightarrow$ Input node, $1 \rightarrow$ Constant node, $2 \rightarrow$ Arithmetic node, $3 \rightarrow$ Test node, $4 \rightarrow$ Output node), g_{i1}, g_{i2} give the associated nodes when $t_i = 2$. g_{i1} gives the input node when $t_i = 3$ or 4 and it is equal to -1 when $t_i \leq 1$. The marks #, (and) are actually real numbers, say -1, -2 and -3.

From the language theory it is known that we can code in $\{0,1\}$ all other words from a finite alphabet. So, if C has s gates, its pseudo-binary description is the word in $\{0,1\}^* \bigcup \{a_1 = 1, ..., a_m\}$ of size bounded by $O(slog(s))$ corresponding to the binary description of all the discrete components of $\phi(C)$ and the description of real constant a_i by itself.

The central property for additive circuits is given in the following lemma which is the circuit version of lemma 1.

Lemma 3 Let z_{it} be the value of the ith gate at depth t of a circuit working on input $x = x_1, ..., x_n$. Then, z_{it} is given in term of the circuit constants $1, a_2, ...a_m$ and of the input $x_1, ..., x_n$ by

$$z_{it} = \sum \alpha_{itj} a_j + \sum \beta_{itj} x_j$$

where $\alpha_{itj}, \beta_{itj}$ are integers satisfying $|\alpha_{itj}|, |\beta_{itj}| \leq 2^t$.

Proof We proceed by induction. In depth 0 the property holds by taking $\beta_{i0i} = 1 (\alpha_{i0i} = 1)$ and $\alpha_{i0j} = \beta_{i0j} = 0, j \geq 0,\ j \neq i$ when the gate is an input one(constant one). Suppose that this is true for $t' < t$. We prove for t. If the operation is an addition or subtraction it is clear that the coefficients α_{it+1j} are bounded by twice the maximum value between α_{itj_1} and α_{itj_2}, where j_1 and j_2 are the input to node i. In the case of test gates we give to α_{i01} the test answer and so the result holds. ■

A family of circuits $C = (c_n)$ is said of polynomial size and polylogarithmic depth(PS-LD) if the circuit c_n has n input nodes and there exist two polynomials p and q such that $p(n)$ is the size of the nth circuit and $q(log(n))$ its depth. The description of a family of circuits C is obtained as

a word being the concatenation of the description of the circuits c_n. So, in the description of C, the description of c_n is written from the m-th position, where m is bounded by

$$m \leq \sum_{s=1}^{n} p(s)(log(p(s))) \leq g(n)$$

where the degree of g is bounded by $degree(P) + 2$.

For a family C we define the language accepted by C as

$$L(C) = \bigcup_{n \geq 1} L(C_n) \quad L(C_n) = \{x \in R^n / c_n(a, x) = 1\}$$

In order to get a uniform model of parallel computation we should endow families of circuits with some uniformity condition. As remark by Cucker in [2], the most usually used in the Boolean case to define the class NC (the generation of the circuits by a machine working in logarithmic space) is meaningless in the real model of computation, since Michaux[6] showed that any recursive subset of R^∞ can be decided by a real Turing machine within linear space. We say that the family C is uniformly generated if there exists an additive machine M working in polynomial time giving the description of the nth circuit when we put as input the sequence $1...1 = 1^n$. We shall assume that the same type of tests is used in the circuits and in the additive machine deciding the uniformity condition. This hypothesis can be discussed which leads to interesting problems not considered in this paper.

We define $NC^{=}_{add}$ as the language class obtained by the union of all the languages accepted by an uniform circuit family PS-LD performing only equality tests. The class $NC^{<}_{add}$ is like $NC^{=}_{add}$ except that the test are $x \geq 0?$. We define also $NNC^{=}_{add}$ and $NNC^{<}_{add}$(non uniform NC^{op}_{add} with $op \in \{=, <\}$) as the union of all the languages accepted by a circuit family PS-LD having tests $x = 0?$ and $x \geq 0?$ respectively.

In order to characterize the class $NNC^{<}_{add}$ in term of $NC^{<}_{add}$ we introduce a new class of complexity. We call $f : I\!N \rightarrow R^\infty$ an advice function. We say that $L \in NC^{op}_{add}/Poly$ if there exists an uniform circuit family PS-LD $C = (c_n)$ and an advice function f, such that $|f(n)|$ is the number of constants in c_n and such that L is recognized by the circuit family C' whose nth circuit is obtained from c_n by changing its constants by the $|f(n)|$ first constants given by $f(n)$.

Proposition 1 $NNC^{<}_{add} = NC^{<}_{add}/Poly$

Proof In order to show the direct inclusion, we prove that any circuit family with constants equal to 1 is uniform. Let $C = (c_n)_{n\geq 0}$ a non-uniform family and let ϕ_n be the binary description of c'_n, the circuit obtained from c_n by setting all its constants to 1. Let ϕ be the concatenation of all the ϕ_n separated by a special symbol $*$ coded in the binary alphabet. Then ϕ is a word which can be coded into a real number α. Using a modification of the machine given in lemma 2 we obtain a machine having α as constant and giving the description of the circuit c'_n in polynomial time and logarithmic space. Thus, the family C' is uniform. Now, we define the advice function f such that $f(n)$ gives the constants of the circuit c_n. So, the family C' with advice f is the family C and the result holds.

The reverse inclusion is derived from the fact that the circuits with a polynomial advice are nonuniform circuits having the advice as constants. ■

3 Restriction to binary inputs

We shall prove that $NC^{=}_{add} \bigcap \{0,1\}^* = NC$ and $NC^{<}_{add} \bigcap \{0,1\}^* = NC/Poly$, where NC is the language class accepted by boolean uniform circuit families of polynomial size and polylogarithmic depth and $NC/Poly$ is obtained from NC by adding an advice function $f : I\!N \rightarrow \{0,1\}^*$ and defining the acceptance like for $NC^{<}_{add}/Poly$. Analogous results for the classes $P^{=}_{add}$ and $NP^{=}_{add}$ were proved by Koiran in [7].

Theorem 1 $NC^{=}_{add} \bigcap \{0,1\}^* = NC$

Proof The direct inclusion is analogous to those given in [7]. We give a sketch of the proof. Let C be an uniform circuit family and let M be the additive machine giving the uniformity of C. Let $a'_1 = 1, a'_2, ..., a'_m$ be its constants and $a_1 = 1, a_2, ..., a_r$ a base of the rational space generated by $1, a'_1, ..., a'_m$. Let b_{ij} be the rational coefficients such that

$$a'_i = \sum_{j=1}^{r} b_{ij} a_j$$

From lemma 1 we know that all the constants of the circuits c_n, $d_1^n = 1, d_2^n, ..., d_{s_n}^n$ can be written as

$$d_k^n = \sum_{i=1}^{m} \lambda_{ki}^n a_i' = \sum_{j=1}^{r} (\sum_{i=1}^{m} \lambda_{ki}^n b_{ij}) a_j$$

$$d_k^n = \sum_{j=1}^{r} \gamma_{kj}^n a_j$$

where $\gamma_{kj}^n \leq 2^{O(p(n))}$. So, the binary description of γ_{kj}^n is a word of size $O(p(n))$. Now, from lemma 3 we know that z_s^n, the value of the s-th gate in circuit c_n, can be written by

$$z_s^n = \sum_{k=1}^{s_n} \alpha_{sk}^n d_k^n$$

Let c_n' be the circuit with constants γ_{kj}^n (in a binary representation) in which each addition or subtraction gate in c_n with input s and s' is replaced by a subcircuit making the addition or subtraction between the binary representation of α_{sk}^n and $\alpha_{s'k}^n$. A test $x = 0?$ is replaced by a subcircuit computing

$$u_{sj} = \sum_{k=1}^{s_n} \alpha_{sk}^n \gamma_{kj}^n$$

for $j = 1, ..., r$ and asking if $u_{sj} = 0$ for all $j = 1, ..., r$. In this case the circuit answers 0 and it answers 1 otherwise. Since the subcircuit can be chosen of polylogarithmic depth and polynomial size, we can obtain a Turing machine with constants b_{ij} giving the description of c_n'(including the constants γ_{kj}^n) working in polynomial time and logarithmic space.

The reverse inclusion is direct because all boolean operations can be executed with the addition and equality tests. ■

Observe that this proof is not true for inequality test because we cannot test $x \geq 0?$ by looking only to the coefficients u_{sj}. Now, we consider the case of inequality tests.

Theorem 2 $NC_{add}^{<} \bigcap \{0,1\}^* = NC/Poly$

Proof Let $C = (c_n)$ in $NC^{<}_{add} \bigcap \{0,1\}^*$ From lemma 3 an input is accepted by C when the constants $a_1, ..., a_m$ satisfy a system of large or strict inequalities(those given by the inequality tests over the variables z_{ik}). We want to change these real constants by rational ones which depend only on n. So, we write a large system describing the evolution of the circuit c_n on all the inputs $x \in \{0,1\}^n$. Since the possible inequalities are finite from theorem 5 in [7] we know that if this system has a solution then it admits a rational solution of size polynomial in the integer coefficients α and β, giving the description of z_{ik} in term of the circuit constants, and in the number of constants. So, the advices for the circuit c_n are the rational solutions of this system.

We prove the reverse inclusion. This proof is rather different from the last theorem. Let $L \in NC/Poly$ then there exists an advice function $f : I\!N \rightarrow \{0,1\}^*$ and an uniform family $C = (c_n)$ such that the circuit c_n obtained by replacing its constants by those of $f(n)$ accepts $L \bigcap R^n$. Let M be the binary Turing machine giving the description of c_n with 1^n as input. We build an additive Turing machine M' having as constant a which is the description of the advice f given by:

$$a = \sum 2^{-j} \lambda_j$$

where the word $\lambda = \phi(f(1))00\phi(f(2))00...00\phi(f(n))00...$ and ϕ is defined by $\phi(0) = 01$ and $\phi(1) = 11$ and its extension to $\{0,1\}^*$ is made in the habitual form. So, making the additions and tests of the form $x \geq 0?$, M' computes the advice associated to the n-th circuit(like the machine given in lemma 2). Coupling the machine M and M' we obtain a machine M'' which gives the description of the n-th circuit of C' in polynomial time using logarithmic space(those used by M) and then $L \in NC^{<}_{add}$. ■

4 $NC_{add} \subsetneq P_{add}$

We prove that there exists a language of $P^{=}_{add}$ which does not belong to $NC^{<}_{add}$. We call EXP the subset of R^∞ defined as follows

$$EXP = \{(x_1, x_2, ..., x_n, 0, ...) \in R^\infty / \forall i \in \{3, ...n\}\ x_i = 1 \wedge 2^n x_1 = x_2\}$$

Lemma 4 $EXP \in P_{add}$

Proof Let M be the additive machine working as follows:

1) It reads x_1 and x_2, and then increment the variable m, which initially has the value 2 until x_n is not equal to 1. If x_n is non zero, then M rejects the input and in other case it continues to step 2.

2) It computes the addition of variable x_1 with itself in y_1, m times. So, y_1 at the end of this process, contains the value $2^n x_1$.

3) It makes the test $y_1 - x_2 = 0?$. M accepts x if the answer is 1 and it rejects otherwise.

Clearly this process takes time $n + n + 1$. ■

Observe that the test $y_1 - x_2 = 0$ can be made by a machine that makes test of the form $x = 0?$ or by a machine that makes test of the form $x \geq 0?$. We have called $P^{=}_{add}$ the first model and $P^{<}_{add}$ the second one when it works in polynomial time[3]. It is clear that $P^{=}_{add} \subset P^{<}_{add}$ and lemma 4 says that EXP belongs to $P^{=}_{add}$.

Theorem 3 *$NC^{=}_{add} \subsetneq P^{=}_{add}$ and $NC^{<}_{add} \subsetneq P^{<}_{add}$*

Proof We prove that EXP$\notin NC^{<}_{add}$. Suppose that there exists a uniform family of additive polylogarithmic circuits recognizing EXP. From lemma 3 we can write the value of any gates p at depth k, $z_{p,k}$ in term of the input variables, $x_1, ..., x_n$ and the constants $a_1 = 1, a_2, ..., a_m$, as follows

$$z_{pk} = \sum_{i=1}^{n} \lambda_{pki} x_i + \sum_{i=1}^{m} \beta_{pki} a_i$$

where $\alpha_{pki}, \beta_{pki}$ are integer bounded by $2^{O(log^R(n))}$ for some R which is a family parameter.

We shall prove that the input $X^n = (2^n x, x, 1^{n-2}, 0, ...)$ and $Y^n = (3^n x, x, 1^{n-2}, 0, ...)$ for $x > 0$ cannot be differentiated by the circuit c_n for n greater enough. It is clear that if two inputs X and Y agree in the circuit tests then X is accepted by c_n iff Y is so. We prove that X^n and Y^n agree in the circuit tests. Suppose that at depth p the gate l has the value x_{pl} (resp. y_{pl}) when the input is X_n(resp. Y_n). By induction we see that x_{pl} and y_{pl} are given by:

$$x_{pl} = \sum \lambda_{plj} X^n_j + \sum \beta_{pli} b_i = \lambda_{pl1} 2^n x + \lambda_{pl2} x + \sum_{j>3} \lambda_{plj} + \sum \beta_{pli} b_i$$

and

$$y_{pl} = \sum \lambda_{plj} Y_j^n + \sum \beta_{pli} b^i = \lambda_{pl1} 3^n x + \lambda_{pl2} x + \sum_{j>3} \lambda_{plj} + \sum \beta_{pli} b_i$$

respectively. Then, if a gate in depth $p+1$ is an addition or subtraction one, it is clear that this description is true in depth $p+1$. Remark that only the sign of λ_{pl1} is important. Indeed, if it is zero then the value for x_{pl} and y_{pl} agree. So, the test gives the same answer. If it is non zero then for n large enough the term $\lambda_{pl1} a^n x$ gives the sign of the test for $a = 2, 3$, and so the answer test is the same for x_{pl} and y_{pl} (if the tests are of the form $x = 0$ all the values x_{pl} and y_{kl} for n big enough are whether equal, when $\lambda_{pl1} = 0$, or non zero). Then the outputs in the test gates agree. Since $X^n \in EXP$ and $Y^n \notin EXP$ we conclude that the language recognized by the family is not EXP. Since the family is an arbitrary one the proof holds. ■

The previous proof can be extended by changing the languages EXP into derived languages. Such an extension has already be done in [2].

Corollary 1 (i) For both test operations the complexity classes defined by parallel time $log^k n$ are different for different k.

(ii) For both test operations the complexity classes defined by parallel time n^k are different for different k.

Proof (i) Consider the languages defined by

$$\{(x_1, x_2, ..., x_n, 0, ...) \in R^\infty / \forall i \in \{3, ...n\}\ x_i = 1 \wedge 2^{log^k n} x_1 = x_2\}$$

and apply the same proof. ■

5 Concluding remarks

In this paper, we have completed the analysis of additive real machines following the work done in [4, 3]. Our main contributions are the definition

of uniform arithmetic circuits, the study of their relationship with standard models when their inputs are binary words, and the proof that $NC_{add} \subsetneq P_{add}$.

The definition of uniformity opens several problems as stated in the paper and should be discussed in particular in view of Michaux' result [6]. We did not introduce the additive PRAM model and hence did not study the relations between the two models. This will be the subject of a subsequent paper.

References

[1] 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 Amer. Math. Soc.*, 21(1):1–46, 1989.

[2] F. Cucker. $\mathrm{P}_R \subsetneq NC_R$. *J. of Complexity*, pages 230–238, 1992.

[3] F. Cucker and M. Matamala. On weak nondeterminism. preprint, May 1993.

[4] P. Koiran. *Puissance de calcul des réseaux de neurones.* PhD thesis, École Normale Sup. de Lyon, June 1993.

[5] K. Meer. A note on a P$\neq$ NP result for a restricted class of real machines. *J. of Complexity*, pages 451–453, 1992.

[6] C. Michaux. Une remarque à propos des machines sur r introduites par blum, shub et smale. *C.R. Acad. Sci. Paris série I, France, 309*, pages 435–437, 1989.

[7] P.Koiran. Computing over the reals with addition and order. preprint, July 1993.