Computability of recurrence equations

Yannick Saouter, Patrice Quinton

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tel.(1) 39 63 55 11

Rapports de Recherche

N° 1203

Programme 2
Structures Nouvelles d'Ordinateurs

COMPUTABILITY OF RECURRENCE EQUATIONS

Yannick SAOUTER
Patrice QUINTON

Avril 1990

RR-1203

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Computability of Recurrence Equations
Calculabilité des équation récurrentes *

Yannick Saouter
Patrice Quinton
IRISA, Campus de Beaulieu, 35042 RENNES-CEDEX, FRANCE

26 février 1990
Publication Interne n° 521 - 28 Pages

Abstract: This paper investigates the computability of recurrence equations. We first recall the results established by Karp et al. on the computability of Systems of Uniform Recurrence Equations, by Rao on Regular Iterative Arrays, and Joinnault's undecidability result on the computability of Conditional Systems of Uniform Recurrence Equations with non bounded domain. Then we consider Systems of Parameterized Affine Recurrence Equations, that is to say, systems of recurrence equations whose domains depend linearly on a size parameter, and we establish that the computability of such systems is also undecidable.

Key-words : systems of affine recurrence equations, systems of uniform recurrence equations, computability, Turing machine, non bounded domains, diophantine equations, parameterized recurrent systems, domains linearly dependant on parameters

Résumé Cet article examine la calculabilité des équations récurrentes. Nous rappelons d'abord les résultats établis par Karp et al. sur la calculabilité des systèmes d'équations récurrentes uniformes, par Rao sur les réseaux itératifs réguliers et le résultat d'indécidabilité de Joinnault sur la calculabilité des systèms d'équations récurrentes conditionnelles à domaine non borné. Puis nous considérons des systèmes d'équations récurrentes affines paramétrés, c'est-à-dire, des systèmes d'équations récurrentes dont le domaine dépend linéairement d'un paramètre de taille, et nous montrons que la calculabilité de tels systèmes est aussi inddependecidable.
Mots-clés : systèmes déquations récurrences affines, systèmes déquations récurrences uniformes, calculabilité, machines de Turing, domaines non bornés, équations diophantienne, systèmes récurrents paramétrés, domaines linéairement dépendant de paramètres

*Recherches partiellement supportées par le PRC C^3 et le projet ESPRIT BRA NANA.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

1 Introduction

Systems of *Uniform Recurrence Equations* were proposed by Karp et al.[7] as a means to derive automatically programs for parallel architectures. Since then, extensions of this formalism were used by many authors, in particular, in the fields of systolic array synthesis. The *computability* of a system of recurrence equations is therefore of primary importance, and is considered as the first point to be examined when trying to implement an algorithm [3, 13]. A system of Recurrence Equations is said to be computable, or also *explicit*, if none of its variable instances depends on itself. The only complete result on this subject concerns the case of *strict uniform equations*: Karp et al. give in [7] a necessary and sufficient condition for such a system of equations to be computable. However, strict uniform recurrence equations are a model too weak to capture the sort of algorithms one usually considers for systolic implementation. In order to broaden the model, several formalisms were introduced. Rao[13] considers *Regular Iterative Algorithms* where the equations are uniform but depend on the index values. He gives a sufficient condition for such a system of equations to be computable, but his computability criterion is very restrictive. Moreover, systems involving broadcasts of values such as the Gaussian elimination[10] or the Toeplitz Solver[4] cannot be directly expressed using RIAs. Wong and Delosme[15] consider *Affine Dependence Algorithms* where the equations involve affine index functions. Rajopadhye[12] introduces *Affine Recurrences Equations*, which is a very similar model. In the general case, Quinton and Van Dongen [11] gives a necessary and sufficient condition for a system of affine recurrence equations to be linearly scheduled. In [16], Yaacobi and Cappello give a sufficient condition for a system of affine recurrence equations to be computable. The method amounts to finding whether the index functions corresponding to cycles of the reduced dependence graph have a stationary point. But this condition is only sufficient, because it does not take into account the domain of the corresponding variable: if the stationary point is outside the domain, there is indeed no computability problem.

In general, the computability of a system of recurrence equation does not amount to finding a linear schedule (see [12, 6] for counterexamples). When the index domain of the system is allowed to be unbounded, Joinnault[6] proved that the computability of the system is undecidable. But as the undecidability is essentially a consequence of the unboundedness hypothesis, this result may be considered as non significant for realistic algorithms, which usually have bounded domains. Indeed, when the index domain is bounded, the problem of the computability of one single system can be solved simply by checking whether the dependence graph is acyclic or not.

However, it was until now an open problem to decide whether a bounded system of recurrence equations whose domain depends linearly on a size parameter is computable or not. This papers investigates this problem, and shows that it is undecidable.

In section 2, we introduce the model of parameterized affine recurrence equations. Then in section 3, we recall the results of Karp et al. on the computability of non conditional uniform recurrence equations, and we present in detail the result of Joinnault about the computability of systems of conditional uniform recurrence equations. In section 4 and 5, we show that the computability of systems of parameterized affine recurrence equations is undecidable.

2 Definitions

The most general type of recurrence equations that we consider is called *systems of parameterized affine recurrence equations.*

A *parameterized affine recurrence equation* is an equation of the form

$$z \in D_p \rightarrow U(z) \quad = \quad f[V(I(z)), ...] \tag{1}$$

where :

- z is a point of $\mathbf{Z}^n$, where $\mathbf{Z}$ denotes the set of integers,
- $p = (p_1, p_2, ..., p_m)$ is a point of $\mathbf{Z}^m$ named *the size parameter* of the equation. We assume that p belongs to a convex polyhedron $\mathbf{P} \subset \mathbf{Z}^m$ (in most cases, $\mathbf{P} = \mathbf{N}^m$, where $\mathbf{N}$ denotes the set of non negative integers).
- D_p is the set of integral points belonging to a convex polyhedron of $\mathbf{Z}^n$, called the *domain* of the equation[1]. We assume that D_p is bounded[2] and is defined by a finite set of linear inequalities involving z and p.
- I is an affine mapping from $\mathbf{Z}^n$ to $\mathbf{Z}^l$ called *index mapping*; I has the form

  $$I(z) = A.z + B.p + C$$

  where the constants A, B and C are integral matrices, $A \in \mathbf{Z}^l \times \mathbf{Z}^n$, $B \in \mathbf{Z}^l \times \mathbf{Z}^m$, and $C \in \mathbf{Z}^l$,
- U and V are *variable names* belonging to a finite set $\mathbf{V}$. Each variable is indexed with an integral index, whose dimension (called the *index dimension* of the variable in the following) is constant for a given variable. The variable $U(z)$ is called the *result* of the equation and $V(I(z))$ is an *argument.*
- f is a single-valued function that depends *strictly* on its arguments; we assume that the function f is *elementary*, i.e. it has complexity $O(1)$ on a Random Access Machine[2].
- the '...' means that there can be other arguments of the same form as $V(I(z))$.
- the domains of two equations having the same variable as result are disjoint. This hypothesis ensures that a variable is not defined twice.

For a given p, equation (1) represents a finite set of *equation instances*, each one of which is associated with a particular point z of D_p. A *system of parameterized affine recurrence equations* (PARE in the following) is a finite set of equations such as (1), having the same

[1]The reader is referred to [14] for an introduction to convex polyhedra.

[2]This hypothesis is not compulsory, but simplifies the following presentation. In particular, it can be useful to assume that the domain has one infinite direction, in order to represent algorithms such as the convolution. All the results in this paper can be extended to cover such a case.

parameter set. Note that all equations need not be indexed in the same subspace, i.e., n is not necessarily the same for all equations.

We define several importants subsets of PARE. A *system of parameterized uniform recurrence equations* (PURE in the following) is a PARE in which all the index functions $I(z)$ are translations. A URE is a PURE without parameters. A URE is said to be *non conditional* if all the equations have the same domain, otherwise, it is *conditional.*

Example 1 The following is a PARE which describes the matrix multiplication of square $n \times n$ matrices $C = AB$. We denote by $c(i,j)$, $a(i,j)$ and $b(i,j)$ the terms i,j of the matrices C, A and B respectively.

$$\begin{aligned} 0 \leq i,j \leq n, k = 0 &\rightarrow C(i,j,k) = 0 \\ 0 \leq i,j \leq n, 0 < k \leq n &\rightarrow C(i,j,k) = C(i,j,k-1) + a(i,k) \times b(k,j) \\ 0 \leq i,j \leq n &\rightarrow c(i,j) = C(i,j,n). \end{aligned}$$

In the following, we shall often write conditional recurrence equations using **case** expression, for example

$$\begin{aligned} &\forall (i,j,k) : 0 \leq i,j,k \leq n \\ &\qquad C(i,j,k) = \begin{cases} \textbf{if } k = 0 \textbf{ then } 0 \\ \textbf{if } k > 0 \textbf{ then } C(i,j,k-1) + a(i,k) \times b(k,j) \end{cases} \\ &\qquad c(i,j) = C(i,j,n). \end{aligned}$$

Provided the conditions of the expression are linear inequalities involving indexes, it is clear that such expressions are equivalent to a PARE.

Given p, we say that $U(x)$ depends strictly on $V(y)$, and we denote $U(x) \succ V(y)$, if there is an equation (1) such that $x \in D_p$ and $y = I(x)$. We denote by $\overset{*}{\succ}$ the transitive closure of $\succ$, and we say that $U(x)$ depends on $V(y)$ if $U(x) \overset{*}{\succ} V(y)$.

The *complete dependence graph* (CDG) of a PARE is defined as follows:

- each instance of a variable $U(z)$, is a node of the CDG,
- there is an edge from $U(z)$ to $V(z')$ if the computation of the variable U at point z depends on the value of the variable V at point z'.

The *reduced dependence graph* (RDG) of a PURE or URE is defined as follows:

- the nodes of this graph are the names of the variables,
- there is an edge from U to V, valued with θ, if for all z in the domain D the computation of $U(z)$ depends on the value of $V(z - \theta(z))$.

Note that this definition cannot be extended immediately to PAREs, as the vector $z - I(z)$ is not necessarily defined.

In the following, we are interested in the *computability* of PAREs. A PARE is said to be computable if there exists a total order on the variable instances which is compatible with the dependence relation, that is to say that, one can define a valuation for every variable instances such that for every edge, in the CDG, from $U(z)$ to $V(z')$, $V(z')$ strictly precedes $U(z)$ in terms of the order of the respective valuations.

In other words, the complete dependence graph of the PARE contains no cycles, and moreover, the set of predecessors of any variable instance is finite.

3 Previous work

3.1 Computability of non conditional UREs

The computability of recurrence equations was considered by Karp et al. [7] and Rao [13]. Karp et al. [7] consider non parameterized non conditional UREs, which have the following form:

$$\forall i, 1 \leq i \leq s, \forall z, z \in D,\ a_i(z) = f_i(a_{i_1}(z), \ldots, a_{i_k}(z)) \tag{2}$$

The important difference with our definition of an URE is that all the equations have the same domain D. In [7], a procedure for deciding whether such a system is computable is presented. It is based on the following theorem:

Theorem 3.1 *A system such as* (2) *is explicitly defined if and only if its associated RDG has no path contained in a nonpositive cycle (i.e. a cycle with a nonpositive valuation).*

The proof of this theorem can be found in [7].

Rao[13] considers Regular Iterative Arrays (RIA), i.e. UREs as they are defined here. The computability analysis he presents is based on the result of Karp's as he considers that all dependences are extended on the whole domain of the system, even if they hold only on a subdomain. As a result, Rao's algorithm gives only a sufficient condition for a URE to be computable. The following example illustrates a case when a URE, which does not satisfy Rao's condition, is however computable.

Example 2 Consider the following equations

$$U(i,j) = \begin{cases} \textbf{if } j > 0 \textbf{ and } 1 \leq i+j \leq n & \textbf{then } U(i,j-1) \\ \textbf{if } j = 0 \textbf{ and } 1 \leq i \leq m & \textbf{then } V(i,j) \\ \textbf{if } i = 0 \textbf{ and } j = 0 & \textbf{then } u \end{cases}$$

$$V(i,j) = \begin{cases} \textbf{if } 1 \leq i < m \textbf{ and } 0 \leq j \leq n & \textbf{then } V(i+1,j+1) \\ \textbf{if } i = m \textbf{ and } 1 \leq j \leq m & \textbf{then } W(i,j) \end{cases}$$

$$W(i,j) = \begin{cases} \textbf{if } i+j > n \textbf{ then } W(i-1,j) \\ \textbf{if } i+j = n \textbf{ then } U(i,j) \end{cases}$$

In this system (see figure 1), each instance $U(p,q)$ with $0 < p$ and $p+q \leq n$ successively depends on: $U(p,0)$, $V(p,0)$, $V(m,m-p)$, $W(m.m-p)$, $W(n-m+p,m-p)$, $U(n-m+p,m-p)$, and $U(n-m+p,0)$.

When $m = n + 1$ (see figure 1(A)), this sequence of derivations is summarized by $U(p, q) \rightarrow U(p-1, 0)$ and then recursively on the first coordinate $U(p, q) \rightarrow U(0, 0) = u$. Therefore the system is computable.

On the other hand, when $m = n$ (figure 1(B)), the derivation can be summarized by $U(p, q) \rightarrow U(p, 0)$, so that each instance $U(p, 0)$ with $p > 0$ depends on itself. The system is not computable.

This example clearly shows that the computability of a conditional system of recurrence equations may depend on the shape of the domain of its equations, and therefore, cannot be checked by looking only at the dependence graph.

3.2 Computability of conditional non-bounded URE

In [6], Joinnault considers the computability of conditional UREs when the domains are not bounded. We recall here her main result, which shows that the problem is undecidable.

The idea is to show that any Turing machine can be encoded as a URE, and that the computability of such a URE implies the decidability of the halting problem of a Turing machine.

Consider a Turing machine, with set of symbols $A = \{a_0, a_1, \ldots, a_{N-1}\}$ and where $a_0 = \flat$ is a special symbol, and set of states Q, initial state q_0, and terminating states $F \subset Q$. The transition function t is defined from $(Q - F) \times A$ to $A \times (L, R) \times Q$, and $t(q_i, a_j) = (a_k, d, q_l)$ means that if the machine is in state q_i and if the cell aimed at by the head contains the symbol a_j, then the head overwrites the cell by the symbol a_k, moves in the direction d and the machine enters state q_l.

At a given instant, the machine is described by a configuration (u, q, a, v) in $A^* \times Q \times A \times A^*$, where u is the word formed by the symbols on the left of the head, q is the current state, a is the letter aimed at by the head and v is the word to the right of the head. A configuration is initial when $u = \emptyset$, $q = q_0$ and $a \neq \flat$. The configuration is said to be final if $u = \emptyset$ and $q \in F$. The following encoding is similar to the encoding of a binary Turing machine ($\flat = 0, A = \{0, 1\}$) described in [9].

Encoding the configurations

- If M is the number of the states in Q, then the states q_i are encoded by $c(q_i) = i$.
- Similarly the N symbols of A are encoded by $c(a_i) = i$, (the special symbol $\flat$ is encoded by 0).
- The word on the left of the head $u = u_0 u_1 \ldots u_k$, in which the code of each $u_i \in A$ is given by $c(u_i)$, is encoded by

$$c(u) = \sum_{i \in [0,k]} c(u_i) N^{k-i}.$$

- Similarly the word on the right of the head $v = v_0 v_1 \ldots v_k$ is encoded by

$$c(v) = \sum_{i \in [0,k]} c(v_i) N^{i}.$$

Thus the machine configurations are entirely described by a 4-tuple $(c(u), c(q), c(a), c(v))$.

Encoding the transitions

- For all pairs (q_i, a_j) linked to another pair (q_k, a_l) by the relation $t(q_i, a_j) = (a_l, R, q_k)$, we write :

$$(u_0 u_1 \dots u_k, q_i, a_j, v_0 v_1 \dots v_l) \vdash^{RT} (u_0 u_1 \dots u_k a_l, q_k, v_0, v_1 \dots v_l).$$

  Let $x = c(u_0 u_1 \dots u_k)$ and $y = c(v_0 v_1 \dots v_l)$. Then $c(u_0 u_1 \dots u_k a_l) = N \times x + l$, and we encode such a transition by means of the following non uniform equation:

$$s(x, i, j, y) = s(N \times x + l, k, Mod(y, N), Div(y, N)).$$

- For all pairs (q_i, a_j) linked to an other pair (q_k, a_l) by the relation $t(q_i, a_j) = (a_l, L, q_k)$, we write :

$$(u_0 \dots u_{k-1} u_k, q_i, a_j, v_0 \dots v_l) \vdash^{LT} (u_0 \dots u_{k-1}, q_k, u_k, a_l v_0 \dots v_l)$$

  which is encoded by the following non uniform equation:

$$s(x, i, j, y) = s(Div(x, N), k, Mod(x, N), N \times y + l)$$

  where x and y are defined as above.

Encoding the Turing machine Misusing the notation t, we abbreviate by $t(i, j) = (i_1, D, j_1)$ when $t(q_i, a_j) = (a_{j_1}, D, q_{j_1})$. With this convention, the machine is encoded by the equations :

$$s(x, i, j, y) = \begin{cases} \textbf{if } i \in c(F) \textbf{ then } RESULT(x, j, y) \\ \textbf{if } t(i, j) = (i_1, R, j_1) \textbf{ then } s(N \times x + j_1, i_1, Mod(y, N), Div(y, N)) \\ \textbf{if } t(i, j) = (i_1, L, j_1) \textbf{ and } x \neq 0 \textbf{ then } s(Div(x, N), i_1, Mod(x, N), N \times y + j_1). \end{cases} \quad (3)$$

The operation of the machine from the initial configuration $(\emptyset, q_0, a, v)$ is realized by the execution of the calculation of $s(0, 0, c(a), c(v))$. Notice that all the conditions such as $t(i, j) = (i_1, d, j_1)$ with $d \in \{R, L\}$ can be expressed as :

$$(i, j) \in t^{-1}(i_1, d, j_1)$$

and $t^{-1}(i_1, d, j_1)$ is a finite union of convex polyhedra. This is true because any such condition defines a point in the encoding space and a point is convex and A and Q are finite sets. Therefore the conditions of equation (3) are compatible with the definition of a PARE. However equation (3) is not a PARE as it involves non-affine functions, such as Div and Mod. The next section shows how to replace this equation by an equivalent PARE.

Uniformization Let us introduce two new variables $RT(x,i,j,y,k)$ and $LT(x,i,j,y,k)$. Our goal is to define new uniform equations in such a way that

$RT(x,i,j,y,0) = s(N \times x + j, i, Mod(y,N), Div(y,N))$,

and

$LT(x,i,j,y,0) = s(Div(x,N), i, Mod(x,N), N \times y + j)$.

Consider the case of variable RT. The idea of the transformation is to use k as an auxiliary variable which serves to compute the quantities $N \times x + j$, $Mod(y,N)$, and $Div(y,N)$.

For example, if we let

$$RT(x,i,j,y,k) = \begin{cases} \textbf{if} \quad x > 0 \quad \textbf{then} \quad RT(x-1,i,j,y,k+N) \\ \textbf{if} \quad x = 0 \quad \textbf{then} \quad RT_1(x,i,j,y,k) \end{cases}$$

then, by induction on x, $RT(x,i,j,y,0) = RT_1(0,i,j,y,N \times x)$. Using the same idea, we introduce the new equations

$$RT_1(x,i,j,y,k) = \begin{cases} \textbf{if} \quad k > 0 \quad \textbf{then} \quad RT_1(x+1,i,j,y,k-1) \\ \textbf{if} \quad k = 0 \quad \textbf{then} \quad RT_2(x,i,j,y) \end{cases}$$

$$RT_2(x,i,j,y) = \begin{cases} \textbf{if} \quad j > 0 \quad \textbf{then} \quad RT_2(x+1,i,j-1,y) \\ \textbf{if} \quad j = 0 \quad \textbf{then} \quad RT_3(x,i,j,y) \end{cases}$$

$$RT_3(x,i,j,y) = \begin{cases} \textbf{if} \quad y > 0 \quad \textbf{then} \quad RT_3(x,i,j+1,y-1) \\ \textbf{if} \quad y = 0 \quad \textbf{then} \quad RT_4(x,i,j,y) \end{cases}$$

$$RT_4(x,i,j,y) = \begin{cases} \textbf{if} \quad j \geq N \quad \textbf{then} \quad RT_4(x,i,j-N,y+1) \\ \textbf{if} \quad j < N \quad \textbf{then} \quad s(x,i,j,y) \end{cases}$$

It is easy to check that

$$\begin{aligned} RT_1(0,i,j,y,N \times x) &= RT_2(N \times x,i,j,y) \\ &= RT_3(N \times x + j,i,0,y) \\ &= RT_4(N \times x + j,i,y,0) \\ &= s(N \times x + j,i,Mod(y,N),Div(y,N)) \end{aligned}$$

The same can be done for $LT(x,i,j,y,0)$. The new system is a PARE, and simulates the functioning of the given Turing machine T.

Let us consider the sublass S of the UREs which are the encoding of a Turing machine. Each URE is a system of the form:

$$X(p) = \begin{cases} \textbf{if} \quad p \in D_1 \quad \textbf{then} \quad f_1(Y_1(p-d_1)) \\ \textbf{if} \quad p \in D_2 \quad \textbf{then} \quad f_2(Y_2(p-d_2)) \\ \vdots \\ \textbf{otherwise} \quad f_s(Y_s(p-d_s)) \end{cases}$$

where all functions f_i are the identity. A computation $X(p)$ terminates if and only if, it depends on a finite number of others computations (as the termination of the functions f_i is certain). Therefore, the problem of the computability of the UREs is undecidable because:

- The computability of the UREs of the subclass S is equivalent to their termination;
- The termination of a URE of the subclass S is equivalent to the termination of the Turing machine it encodes.

We have therefore the following result:

Proposition 3.1 *The problem of the computability of the URE is undecidable.*

To conclude, let us notice that URE belonging to class S are defined on unbounded domains, since the maxima of the variables x and y are not a priori known. As we have already seen, the computability of URE defined over bounded domains is decidable. Therefore if those maxima were computable, the general problem would be itself decidable which is in contradiction with Proposition 3.1. The difficulty lies therefore in the fact that the domain is unbounded, and this result seems irrelevant for practical implementations of systolic-like architectures. In the following, we shall show that even when the domains of the system are bounded, the difficulty is still present, if we consider that the domains depend on a size parameter.

4 Linearly dependent PAREs

From now on, we are interested in PAREs, that is to say, system of recurrence equations whose domains depends linearly on one or several parameters. In this section, we prove that the computability of PAREs involves the tenth Hilbert's problem, which is known to be undecidable [5]. Paragraph 4.1 presents an example. In paragraph 4.2, we introduce mathematical notions which are needed, and we give the undecidability result, which is proved in section 5.

4.1 An illustrative example

In order to show that the problem of determining the computability of bounded conditional PAREs is intrinsically difficult, we consider the following example.

Example 3 Let $D = \{(p,k) \mid 1 \leq p \leq n, 1 \leq k \leq n-1\}$. Consider the following PARE over D parameterized by n:

$$Prime(p,k) = \begin{cases} \textbf{if } k > 1 \textbf{ then } \neg Multiple(p,k) \wedge Prime(p,k-1) \\ \textbf{if } k = 1 \textbf{ then true} \end{cases} \quad (4)$$

$$Multiple(p,k) = \begin{cases} \textbf{if } p > k \textbf{ then } Multiple(p-k,k) \\ \textbf{if } p = k \textbf{ then } Prime(n,n-1)\ (y) \\ \textbf{if } 0 < p < k \textbf{ then false} \end{cases} \quad (5)$$

This PARE defines two boolean variables. $Multiple(p,k)$ is true if and only if p is a multiple of k, and $Prime(p,k)$ is true if and only if p is a prime number, independently on k. The complete dependence graphs of the system when n=5 and n=6 are shown respectively in figures 3 and 2.

Let us show that this PARE is computable if and only if n is a prime number. Indeed according to equation (4) we have:

$$Prime(n, n-1) = \neg Multiple(n, n-1) \wedge \neg Multiple(n, n-2) \wedge \ldots \wedge \neg Multiple(n, 2)$$

Two cases can be considered:

- if $n = pq$, then $Prime(n, n-1)$ depends on $Multiple(n, p)$. But according to equation (5), $Multiple(pq, p) = Multiple(p(q-1), p) = ... = Multiple(p, p) = Prime(n, n-1)$. Therefore $Prime(n, n-1)$ depends on itself and the system is not computable.
- if n is a prime number, the calculus of $Prime(n, n-1)$ never leads to a term $Multiple(p, p)$. Therefore $Prime(n, n-1)$ is computable. Moreover, the calculus of a term $Prime(p, k)$ always leads to terms with decreasing value of $p + k$. As a consequence, the system is computable.

4.2 Some results about diophantine equations

Let $\mathbf{Z}[X_1, X_2, \ldots, X_k]$ denote the ring of polynomials of k unknowns over $\mathbf{Z}$. We consider diophantine equations, that is equations of the form $Q[x_1, x_2, \ldots, x_k] = 0$ where $Q \in \mathbf{Z}[X_1, X_2, \ldots, X_k]$. Let us recall the following result whose proof can be found in [8]:

Theorem 4.1 *Determining whether a diophantine equation has a integral nonnegative solution is undecidable.*

Let P be a polynomial of $\mathbf{Z}[X_1, X_2, \ldots, X_k]$. We say that P is *lexicographical* when either:

- $P = 0$
- $P = Q_1 + Q_2$ where $Q_1 \in \mathbf{Z}[X_1, \ldots, X_s]$, $Q_2 \in \mathbf{Z}[X_{s+1}, \ldots, X_k]$, Q_2 is lexicographical and $Q_1[X_1, \ldots, X_s] = \pm X_1 \times \ldots \times X_s$

The set of lexicographical polynomials will be denoted by $\mathbf{L}$ in the following.

Given a lexicographical polynomial P, we define P^+ (resp. P^-) as the sum of the positive (resp. negative) monomials of P. Note that $P = P^+ - P^-$. We denote by $n(P)$ the number of monomials of P. Any equation $P = 0$ is called a *lexicographical diophantine equation.* For each diophantine equation $Q = 0$, there exists obviously a lexicographical diophantine equation, $P = 0$, and a set of equalities which are satisfied if and only if $Q = 0$ is satisfied. Therefore, we have the following theorem which results directly from Theorem 4.1

Theorem 4.2 *Given a lexicographical polynomial and a finite set of equalities involving its variables, the problem of determining whether there exists a tuple of nonnegative integers which satisfies the equalities and zeroes out the polynomial is undecidable.*

The following example illustrates our notations.

Example 4 Let $Q[X_1, X_2] = {X_1}^2 + 3X_1X_2 - 2{X_2}^2 + 1$ and let $P(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9) = p_1p_2 + p_3p_4p_5 - p_6p_7p_8 + p_9$. The set of the solutions of $Q[X_1, X_2] = 0$ is the set of the solutions of $P(p_1, \ldots, p_9) = 0$ together with the following additional equations :

$$\{p_1 = p_2 = p_4,\ p_5 = p_7 = p_8,\ p_3 = 3,\ p_6 = 2,\ p_9 = 1\}.$$

We shall also need the following result:

Theorem 4.3 *Given $Q \in \mathbf{L}$, determining whether the equation $Q[X_1, \ldots, X_k] = n(Q^+) - n(Q^-)$ with $Q \in \mathbf{L}$ restricted by a set of equalities has an integral non-negative solution is an undecidable problem.*

Proof: Let $Q[X_1, \ldots, X_k] \in \mathbf{L}$ and let:

$$Q'[X_1, \ldots, X_k, Y_1, \ldots, Y_r, Z_1, \ldots, Z_s] = Q[X_1, \ldots, X_k] - Y_1 - \ldots - Y_r + Z_1 + \ldots + Z_s.$$

Then $n(Q'^+) = n(Q^+) + s$ and $n(Q'^-) = n(Q^-) + r$. Moreover

$$Q'[X_1, \ldots, X_k, 2, \ldots, 2] = Q[X_1, \ldots, X_k] - 2r + 2s.$$

Therefore

$$Q'[X_1, \ldots, X_k, 2, \ldots, 2] = n(Q'^+) - n(Q'^-) \tag{6}$$

is equivalent to

$$Q[X_1, \ldots, X_k] = n(Q^+) - n(Q^-) + r - s.$$

If we let $r = n(Q^-)$ and $s = n(Q^+)$ then (6) is equivalent to $Q[X_1, \ldots, X_k] = 0$. So for each lexicographical diophantine equation constrained with a set of equalities there exists at least one equation of the form $Q'[X_1, ..., X_l] = n(Q'^+) - n(Q'^-)$ with its own set of constraints such that the two equations are equivalent. Whence the result, according to Theorem 4.2. ■

5 Undecidability of the computability of PAREs

The remaining of this paper is devoted to proving the following result:

Theorem 5.1 *Given $Q \in \mathbf{Z}[\mathrm{X}_1, \ldots, \mathrm{X}_k]$ and $Q \in \mathbf{L}$, there exists a PARE whose parameters are $(N_1, \ldots, N_k)$ and which is computable if and only if $Q[N_1, \ldots, N_k] \neq n(Q^+) - n(Q^-)$.*

If we assume this theorem to be true then, obviously:

Corollary 5.1 *Given a PARE, it is undecidable to determine whether there exists values of its parameters, such that the instance of the system is not computable.*

5.1 Informal explanation of the proof

The underlying idea of the proof is the following one. In a m-dimensional hyperparallelepiped defined by the inequalities $\forall\ i \in [1..m]\ :\ 1 \leq p_i \leq N_i$, there are $\prod_{j=1}^{j=m} p_j$ integral points. Therefore, if we cover the hyperparallelepiped in lexicographical order, $\prod N_j - 1$ steps are needed to reach every integral point. Let us call *extremal point* of the hyperparallelepiped the point whose coordinates are maximum, i.e. $(N_1, ..., N_m)$ and let $\mathbf{1}_m$ denote the m-tuple $(1, 1, ..., 1)$. If we cover simultaneously another hyperparallelepiped using other coordinates, then from the extremal point of the domain we will reach $\mathbf{1}_m$ if and only if the number of integral points of both hyperparallelepipeds are equal. The idea is thus to define a PARE whose dependence graph covers both hyperparallelepipeds, and which contains an arc between the point $\mathbf{1}_m$ and the extremal point. This PARE will be computable if and only if the number of integral points of both hyperparallelepipeds are identical.

This process makes it possible to solve the case of polynomials of the form $A - B$ where A and B are lexicographical monomials. The general case of the lexicographical polynomials is handled by dealing serially with all the positive monomials on one hand and all the negative monomials on the other hand.

Consider for example $Q = p_1.p_2 - p_3$. Let $D = \{(i, j, k) \mid 1 \leq i \leq N_1, 1 \leq j \leq N_2, 1 \leq k \leq N_3\}$ a three-dimensional domain. Figure 4 illustrates this example. The extremal point of the domain is $A = (N_1, N_2, N_3)$. From A, we go to $B = (N_1, N_2 - 1, N_3)$, which is one step of the covering of the hyperparallelepiped $H1 = \{1 \leq i \leq N_1, 1 \leq j \leq N_2\}$. Then we make one step on the second hyperparallelepiped $H2 = \{1 \leq k \leq N_3\}$, reaching $a = (N_1, N_2 - 1, N_3 - 1)$. The covering proceeds on by alternatively moving one step on each hyperparallelepiped, which gives the path $A \mapsto B \mapsto a \mapsto C \mapsto b \mapsto D \mapsto c \mapsto E \mapsto d \mapsto F \mapsto e$ until we finally reach the point $e = (N_1, 1, N_3 - N_2 + 1)$. To reach the next point G for the covering of $H1$, we must decrease the first coordinate while the second one is set to N_2. So the coordinates of G are $(N_1 - 1, N_2, N_3 - N_2 + 1)$. The next step is $G \mapsto f$, etc. This principle leads to a PARE which is computable if and only if $N_1.N_2 - N_3 = 0$. Indeed, in a lexicographical covering the conditions for reaching the edge of the hyperparallelepiped are linear. We differentiate the covering of H_1 and H_2 by defining two variables X_1 and X_2.

More generally, given a lexicographical polynomial, once its positive and negative part are separated, we build a PARE which is computable if and only if $Q(N_1, ..., N_k) \neq n(Q^+) - n(Q^-)$ (the terms $n(Q^+)$ and $n(Q^-)$ appears because of the term -1).

5.2 Notations

Let us introduce some notations in order to make the proof easier to follow. We will denote by $\mathbf{1}_k$ the t-uple (1,1,...,1) and we let $\mathbf{1}_0 = \emptyset$. Let $Q[p_1, \ldots, p_k]$ be a lexicographical polynomial. Without loss of generality, we assume that Q is written $Q = \sum_{a=1}^{r}(M_a) - \sum_{b=1}^{t}(m_b)$ where M_a and m_b are the positive and negative monomials of Q, and moreover, that the letters of the successive monomials are $p_1, ..., p_k$ in this order. Let $P = (p_1, ..., p_k)$. With this convention, any monomial M_a (resp. m_b) corresponds to a unique subsequence $p_i, p_{i+1}, ..., p_{i+m}$ of P as

$M_a = p_i.p_{i+1}...p_{i+m}$. Given M_a, we let $P = (P^+_{a]}, P^+_a, P^+_{[a})$, where

$$\begin{aligned} P^+_{a]} &= (p_1, ..., p_{i-1}) \\ P^+_a &= (p_i, ..., p_{i+m}) \\ P^+_{[a} &= (p_{i+m+1}, ..., p_k). \end{aligned}$$

In other words, P^+_a denotes the subsequence of P corresponding to M_a, and $P^+_{a]}$ and $P^+_{[a}$ denote respectively the prefix and suffix of P^+_a in P. Finally, we let $N^+_{a,l} = (N_i, ..., N_{i+l-1})$. Similar notations ($P^-_{b]}$, P^-_b, $P_{[b}$), and $N^-_{b,l}$ hold for negative monomials m_b. Finally, given a sequence $P = (p_1, ..., p_k)$, we denote by $\mid P \mid$ its length.

Example 5 Let

$$Q[p_1, ..., p_9] = p_1p_2 + p_3p_4p_5 + p_6 - p_7p_8p_9.$$

Then

$$\begin{aligned} \{M_a\} &= \{p_1p_2, p_3p_4p_5, p_6\} \\ \{m_b\} &= \{p_7p_8p_9\} \\ P &= (p_1, ..., p_9) \\ P^+_2 &= (p_3, p_4, p_5) \\ P^+_{[2} &= (p_1, p_2) \\ P^+_{2]} &= (p_6, p_7, p_8, p_9) \end{aligned}$$

5.3 System associated with a lexicographical polynomial

Let $Q[p_1, ..., p_k] = \Sigma_{a=1}^r M_a - \Sigma_{b=1}^t m_b$ be a lexicographical polynomial. We associate with Q a PARE over the domain $\{ (p_1, \ldots, p_k) \mid \forall i \in \{1, ..., k\}\ 1 \leq p_i \leq N_i \}$, defined as follows:

$$X(P) = \left\{ \begin{array}{lll} \textbf{if} \quad P = (N_1, ..., N_k) & \textbf{then} & X_{1,1}(P) \\ & \textbf{otherwise} & \textbf{constant} \end{array} \right. \tag{7}$$

$\forall\ a, b\ :\ (1 \leq a \leq r \wedge 1 \leq b \leq t)$

$$X_{a,b}(P) = \left\{ \begin{array}{l} \textbf{if } a = r,\ b = t \textbf{ and } P = \mathbf{1}^k \textbf{ then } X(N_1, ..., N_k) \\ \textbf{else} \left\{ \begin{array}{lll} \textbf{if} \quad P^+_a = \mathbf{1}_{\mid P^+_a \mid} & \textbf{then} & X_{a+1,b}(P) \\ \textbf{if} \quad P^+_a = (1, x, u) \textbf{ and } x \neq 1 & \textbf{then} & Y_{a,b}(P^+_{a]}, N^+_{a,1}, x-1, u, P^+_{[a}) \\ \vdots & & \\ \textbf{if} \quad P^+_a = (\mathbf{1}_l, x, u) \textbf{ and } x \neq 1 & \textbf{then} & Y_{a,b}(P^+_{a]}, N^+_{a,l}, x-1, u, P^+_{[a}) \\ \vdots & & \end{array} \right. \end{array} \right. \tag{8}$$

$$Y_{a,b}(P) = \left\{ \begin{array}{l} \textbf{if } a = r,\ b = t \textbf{ and } P = \mathbf{1}^k \textbf{ then constant} \\ \textbf{else} \left\{ \begin{array}{lll} \textbf{if} & P_b^- = \mathbf{1}_{|P_b^-|} & \textbf{then } Y_{a,b+1}(P) \\ \textbf{if} & P_b^- = (1, x, u) \textbf{ and } x \neq 1 & \textbf{then } X_{a,b}(P_{b]}^-, N_{b,1}^-, x-1, u, P_{[b}^-) \\ \vdots & & \\ \textbf{if} & P_b^- = (1_l, x, u) \textbf{ and } x \neq 1 & \textbf{then } X_{a,b}(P_{b]}^-, N_{b,l}^-, x-1, u, P_{[b}^-) \\ \vdots & & \end{array} \right. \end{array} \right. \quad (9)$$

If $a = r+1$ or $b = t+1$ then :

$$\begin{aligned} X_{a,b}(P) &= \textbf{constant} \\ Y_{a,b}(P) &= \textbf{constant} \end{aligned} \quad (10)$$

A few remarks are in order.

- The use of the keyword **else** implies a priori non-convex domains. But these domains are finite unions of convex domains, which is not in contradiction with our definition of a PARE. The use of the keyword **constant** is also not in contradiction with the definition of a PARE's since a constant is a variable with 0 dimension.
- Indices a and b of the system should be interpreted as indexes upon respectively $\{M_l\}$ and $\{m_l\}$. Variables X are used to cover the first hyperparallelepiped, and variables Y to cover the second one. So, intuitively, the definition of $X_{a,b}$ means: if the coordinates related to M_a are all equal to 1, then count with the next monomial, (first alternative of equation (8)), and if not, subtract 1 (lexicographically speaking) from the coordinates corresponding to M_a and count on the second hyperparallelepiped (second alternative of equation (8)).

Example 6 Consider again the polynomial $Q[p_1, ..., p_9] = p_1p_2 + p_3p_4p_5 + p_6 - p_7p_8p_9$. To Q corresponds the following PARE:

$$\begin{aligned} X(p_1,p_2,p_3) &= \left\{ \begin{array}{l} \textbf{if } (p_1 = N_1, p_2 = N_2, p_3 = N_3) \textbf{ then } X_1(p_1,p_2,p_3) \\ \quad \textbf{else constant} \end{array} \right. \\ X_{1,1}(p_1,p_2,p_3) &= \left\{ \begin{array}{l} \textbf{if } (p_1 = p_2 = p_3 = 1) \textbf{ then } X(N_1,N_2,N_3) \\ \quad \textbf{else if } ((p_1,p_2) = (1,1)) \textbf{ then } X_{2,1}(p_1,p_2,p_3) \\ \quad\quad \textbf{else if } (p_2 > 1) \textbf{ then } Y_{1,1}(p_1,p_2-1,p_3) \\ \quad\quad\quad \textbf{else } Y_{1,1}(p_1-1,N_2,p_3) \end{array} \right. \\ Y_{1,1}(p_1,p_2,p_3) &= \left\{ \begin{array}{l} \textbf{if } (p_1 = p_2 = p_3 = 1) \textbf{ then constant} \\ \quad \textbf{else if } (p_3 = 1) \textbf{ then } Y_{1,2}(p_1,p_2,p_3) \\ \quad\quad \textbf{else } X_{1,1}(p_1,p_2,p_3-1) \end{array} \right. \\ X_{2,1}(p_1,p_2,p_3) &= \textbf{constant} \\ Y_{1,2}(p_1,p_2,p_3) &= \textbf{constant} \end{aligned}$$

As we shall see, this PARE is computable if and only if $N_1.N_2 - N_3 \neq 0$. Indeed, if we consider the path of derivations of $X(N_1, N_2, N_3)$ three cases may occur:

- Case 1: the path ends up at point $(1,1,1)$ and the last step is made on $H2$. Then all the points of $H1$ have been covered and all points of $H2$ but the extremal one have also been

covered. Moreover the number of steps on the two hyperparallelepipeds are equal, that is to say $N_1.N_2 - 1 = N_3 - 1$. Therefore, $Q(N_1, N_2, N_3) = 0 = n(Q^+) - n(Q^-)$.

- Case 2: the path does not end at point (1,1,1). The last point reached by the path, say x, has one of the following forms:
  - $x = (1, 1, p_3)$, with $p_3 > 1$. Then, either the number of steps on both hyperparallelepiped are equal if the last step was on $H2$, or one more step has been made on $H1$. Thus $N_1.N_2 - 1 = N_3 - p_3 - 1 + \delta$, where $\delta \in \{0, 1\}$. It follows that $Q(N_1, N_2, N_3) = -p_3 + \delta$. As $p_3 > 1$, $Q(N_1, N_2, N_3) \neq 0$ and $Q(N_1, N_2, N_3) \neq n(Q^+) - n(Q^-)$.
  - $x = (p_1, p_2, 1)$, with $p_1 > 1$ or $p_2 > 1$. Then $N_2(N_1 - p_1 - 1) + (N_2 - p_2) - 1 = N_3 - 1 + \delta)$, $\delta \in \{O, 1\}$, so that $Q(N_1, N_2, N_3) = N_2.p_1 + p_2 + (1 \text{ or } 0)$. But $p_1 > 1$ or $p_2 > 1$, thus $Q(N_1, N_2, N_3) > 0$ and $Q(N_1, N_2, N_3) > 0$.
- Case 3: the path ends up at point (1,1,1) but the last step has been made on $H1$. In this case, one more step was made on $H1$, so that $N_1.N_2 - 1 = N_3 - 1 + 1$. Again $Q(N_1, N_2, N_3) \neq n(Q^+) - n(Q^-)$.

In order to prove theorem 5.1, we will show that the above defined PARE is computable if and only if $Q(N_1, ..., N_k) \neq n(Q^+) - n(Q^-)$. To a monomial $P = p_h p_{h+1} ... p_f$, let us associate a polynomial $\Lambda(P)$ defined by

$$\Lambda(P) = (p_h - 1) + N_h(p_{h+1} - 1) + ... + N_h N_{h+1} ... N_{f-1}(p_f - 1).$$

Define

$$F(U, p_1, \ldots, p_k) = \lambda(U) + \sum_{a=1}^{r} \Lambda(M_a)(P_a^+) - \sum_{b=1}^{t} \Lambda(m_b)(P_b^-)$$

where $\lambda(U) = 0$ if U is a X variable and $\lambda(U) = 1$ if U is a Y variable. Finally, let

$$G(p_1, \ldots, p_k) = \sum_{a=1}^{r} \Lambda(M_a)(P_a^+) + \sum_{b=1}^{t} \Lambda(m_b)(P_b^-)$$

Misusing the notations in order to make them lighter, we will often write expressions such as $\Lambda(M_a)(p_1, ..., p_k)$ or $\Lambda(M_a)(U, p_1, ..., p_k)$ which will have to be understood as $\Lambda(M_a)(P_a^+)$.

To establish the result, we show that F is constant along the derivations and that G decreases along the derivations but is positive. Intuitively, F represents a number that is approximately equal to the difference of the steps made on each hyperparallelepiped and G is a function which ensures that the index is reaching $\mathbf{1}_k$ since G is strictly decreasing along each one-step derivation ang G is a positive function which is null only at point $\mathbf{1}_k$ as we will show further.

First of all, we shall need a technical lemma :

Lemma 5.1 *If* $P = p_h p_{h+1} ... p_f$ *then if* $o < f$ *:*

$$\Lambda(P)(N_h, ..., N_o, x_1, x_2, ..., x_{f-o}) + 1 = \Lambda(P)(1, ..., 1, x_1 + 1, x_2, ..., x_{f-o}) \quad (11)$$

$$\Lambda(P)(N_h, ..., N_f) = N_h N_{h+1} ... N_f \quad (12)$$

Proof : For all $o < f$ and for any x_1, we have :

$$[\ 1\ +\ (N_h - 1) + N_h(N_{h+1} - 1) + ... + N_h N_{h+1}...N_o(x_1 - 1)\ ] =$$
$$[\ N_h + N_h(N_{h+1} - 1) + ... + N_h N_{h+1}...N_o(x_1 - 1)\ ] =$$
$$N_h\ [\ 1\ +\ (N_{h+1} - 1) + ... + N_{h+1}...N_o(x_1 - 1)\ ] =$$
$$N_h N_{h+1}\ [\ 1\ +\ (N_{h+2} - 1) + ... + N_{h+1}...N_o(x_1 - 1)\ ] =$$
$$\vdots$$
$$N_h N_{h+1}...N_o x_1$$

So we can already prove (12). Indeed if we force $o = f - 1$ and $x_1 = N_f$, the previous equality establishes (12). Moreover, we have in the general case :

$$\Lambda(P)(N_h, ..., N_o, x_1, x_2, ..., x_{f-o}) + 1\ =$$
$$[\ 1\ +\ (N_h - 1) + N_h(N_{h+1} - 1) + ... + N_h N_{h+1}...N_o(x_1 - 1)\ ] +$$
$$N_h N_{h+1}...N_o N_{o+1}(x_2 - 1) + ...N_h N_{h+1}...N_{f-1}N_f(x_{f-o} - 1) =$$
$$N_h N_{h+1}...N_o x_1 + N_h N_{h+1}...N_o N_{o+1}(x_2 - 1) + ...N_h N_{h+1}...N_{f-1}N_f(x_{f-o} - 1) =$$
$$\Lambda(P)(1, ..., 1, x_1 + 1, x_2, ..., x_{f-o})$$

which proves (11). ■

Lemma 5.2 *The following properties hold:*

$$G(x) \geq 0, \forall x \in \mathbf{N}^k \tag{13}$$
$$G(x) = 0 \Longleftrightarrow x = \mathbf{1}_k \tag{14}$$
$$F(X, 1, \ldots, 1) = 0 \tag{15}$$
$$F(X, N_1, \ldots, N_k) = Q(N_1, \ldots, N_k) - n(Q^+) + n(Q^-). \tag{16}$$

Proof : Properties (13), (14) and (15) are obvious. Property (16) is obtained by recurrence upon the equalities in Lemma 5.1:

$$\begin{aligned} F(X, N_1, ..., N_k) &= \Sigma_{a=1}^r \Lambda(M_a)(X, N_1, ..., N_k) - \Sigma_{b=1}^t \Lambda(m_b)(X, N_1, ..., N_k) \\ &= \Sigma_{a=1}^r \{1 + \Lambda(M_a)(X, N_1, ..., N_k)\} - \\ &\quad \Sigma_{b=1}^t \{1 + \Lambda(m_b)(X, N_1, ..., N_k)\} - r + t \end{aligned} \tag{17}$$

But we have $r = n(Q^+)$ and $t = n(Q^-)$. Moreover the variables involved in the terms $\Lambda(M_a)(X, N_1, ..., N_k)$ is the set $N_a^+ = (N_i, ..., N_{i+m})$ when $M_a^+ = (p_i, ..., p_{i+m})$. So we have :

$$\begin{aligned} &\Sigma_{a=1}^r \{1 + \Lambda(M_a)(X, N_1, ..., N_k)\} - \Sigma_{b=1}^t \{1 + \Lambda(m_b)(X, N_1, ..., N_k)\} \\ &\quad = \Sigma_{a=1}^r \prod_{N_i \in N_a^+} N_i - \Sigma_{b=1}^t \prod_{N_i \in N_b^-} N_i = Q(N_1, ..., N_k) \end{aligned} \tag{18}$$

according to (12). So (17) and (18) imply :

$$F(X, N_1, ..., N_k) = Q(N_1, ..., N_k) - n(Q^+) + n(Q^-) \quad (19)$$

■

Lemma 5.3 *If* $U(p_1, p_2, \ldots, p_k)$ *depends on* $V(p'_1, p'_2, \ldots, p'_k)$ *in the PARE then:*

$$F(U, p_1, p_2, \ldots, p_k) = F(V, p'_1, p'_2, \ldots, p'_k).$$

Proof : We prove this proposition by induction on the number of times an equation is rewritten. As a basis let us consider a direct dependence, i.e. a one-step dependence. Sequentially speaking, a direct dependence of $X_{a,b}(P)$ consists in a lexicographical subtraction over the coordinates M_a and m_b. If all the coordinates of the first set are equal to 1, then the direct dependence leads to $X_{a+1,b}(p_1, ..., p_k)$. Thus the coordinates do not change and neither does the value of F along this direct dependence. If $P_a^+ = \mathbf{1}_m, x, ..$ with $x \neq 1$ then the direct dependence leads to :

$$Y_{a,b}(u, N^+_{a,m}, x - 1, ...).$$

Thus

$$\begin{aligned} &F(X, p_1, \ldots, p_k) - F(Y, p'_1, \ldots, p'_k) = \\ &\quad \lambda(X) + \Sigma^r_{a'=1}\Lambda(M'_a)(p_1, ..., p_k) - \Sigma^t_{b'=1}\Lambda(m'_b)(p_1, ..., p_k) \\ &\quad -\lambda(Y) - \Sigma^r_{a'=1}\Lambda(M'_a)(p'_1, ..., p'_k) + \Sigma^t_{b'=1}\Lambda(m'_b)(p'_1, ..., p'_k) \end{aligned}$$

according to the definitions of F. The index values P and P' differs only on the set P_a^+, so the terms $\Lambda(M'_a)(p_1, ..., p_k)$ and $\Lambda(M'_a)(p'_1, ..., p'_k)$ are equal as soon as $a' \neq a$. Moreover, the terms $\Lambda(m'_b)(p_1, ..., p_k)$ and $\Lambda(m'_b)(p'_1, ..., p'_k)$ are always equal. At last we have $\lambda(X) = 0$ and $\lambda(Y) = 1$, so :

$$F(X, p_1, \ldots, p_k) - F(Y, p'_1, \ldots, p'_k) = \Lambda(M_a)P_a^+ - \Lambda(M_a)P'^+_a - 1 \quad (20)$$

But $P'^+_a = (N^+_{a,m}, x - 1, ...)$, so according to (11) :

$$\begin{aligned} \Lambda(M_a)P'^+_a + 1 &= \Lambda(M_a)(\mathbf{1}_m, x, ...) \\ &= \Lambda(M_a)(P_a^+) \end{aligned} \quad (21)$$

Then we have :

$$F(X, p_1, \ldots, p_k) - F(Y, p'_1, \ldots, p'_k) = 0 \quad (22)$$

The same can be done with Y, hence the proposition. ■

Proposition 5.1 *If* $U(p_1, \ldots, p_k)$ *depends on* $V(p'_1, \ldots, p'_k)$ *then* $G(p_1, \ldots, p_k) > G(p'_1, \ldots, p'_k)$.

Proof: This proposition is established by induction on the number of direct dependences. The proof is identical to that of Lemma 5.3 except that the term -1 coming from the occurrences of λ in the definition of F do not exist in the definition of G. So the result of the subtraction of $G(p_1, \ldots, p_k)$ and $G(p'_1, \ldots, p'_k)$ along a one-step derivation is equal to 1. ■

Proof of the Theorem Given these lemmas, we are able to prove the theorem. Indeed at one step of direct dependence, one of the following cases arises:

$$X_{a,b}(p_1,\ldots,p_k) = \begin{cases} X_{a+1,b}(p_1,\ldots,p_k) \\ \text{or} \\ X_{a,b+1}(p_1,\ldots,p_k) \\ \text{or} \\ Y_{a,b}(p'_1,\ldots,p'_k) \quad \textit{with } G(p'_1,\ldots,p'_k) < G(p_1,\ldots,p_k) \end{cases}$$

But $G \geq 0$ (lemma 5.2), and G constitutes a well formed order. By induction on the direct dependences we have one of the following cases:

$$X_{a,b}(p_1,\ldots,p_k) = \begin{cases} U_{a+1,b}(p'_1,\ldots,p'_k) \\ \text{or} \\ U_{a,b+1}(p'_1,\ldots,p'_k) \end{cases}$$

where $U \in \{X, Y\}$.
Then by induction on a and b:

$$X_{a,b}(p_1,\ldots,p_k) = \begin{cases} U_{r+1,b'}(p'_1,\ldots,p'_k) & \text{with } b' \neq t+1 \\ \text{or} \\ U_{a',t+1}(p'_1,\ldots,p'_k) & \text{with } a' \neq r+1 \\ \text{or} \\ U_{r,t}(p'_1,\ldots,p'_k) \end{cases} \tag{23}$$

The computation of $X(N_1,\ldots,N_k)$ leads to an infinite loop if and only if we are in the third case with $U = X$ and $(p'_1,\ldots,p'_k) = (1,\ldots,1)$. As F is constant along the direct dependences, $F(X,N_1,\ldots,N_k,1) = F(X,1,\ldots,1)$, which implies that $Q(N_1,\ldots,N_k) = n(Q^+) - n(Q^-)$ because of (16) and (15). If the computation does not loop, we are either in the first or second case.

- in the first case, we have $P'^M = \mathbf{1}_M$ and $U = X$. If $F(X,p'_1,\ldots,p'_k)$ was equal to 0, then we would have:

  $$p'_1 = \ldots = p'_k = 1$$

  which is impossible as the computation of $X(N_1,\ldots,N_k)$ cannot loop. So according to (16), this implies $Q(N_1,\ldots,N_k) \neq n(Q^+) - n(Q^-)$.

- in the second case, we have $P'^m = \mathbf{1}_m$ and $U = Y$ then :

  $$F(X,N_1,\ldots,N_k,1) = F(Y,p'_1,\ldots,p'_k) > 1$$

  i.e. not equal to 0, so again $Q(N_1,\ldots,N_k) \neq n(Q^+) - n(Q^-)$.

So if the computation of $X(N_1,...,N_k)$ is assumed not to loop, then from what precedes we must have $Q(N_1,\ldots,N_k) = n(Q^+) - n(Q^-)$.

On the other hand if this computation is assumed not to loop then $Q(N_1, \ldots, N_k) \neq n(Q^+) - n(Q^-)$

We can now establish the theorem. Suppose first that $Q(N_1, ..., N_k) = n(Q^+) - n(Q^-)$ and the associated system does not loop. This implies that the computation of $X(N_1, ..., N_k)$ does not loop and thus, that $Q(N_1, \ldots, N_k) \neq n(Q^+) - n(Q^-)$, which is a contradiction. As a consequence, if $Q(N_1, \ldots, N_k) = n(Q^+) - n(Q^-)$, then the associated system is not computable.

On the other hand, if $Q(N_1, \ldots, N_k) \neq n(Q^+) - n(Q^-)$ and if the system is assumed to loop, then any non-computable variable leads by derivation to $X(N_1, ..., N_k)$ using the third case of equation (23), as we have just seen. As the right hand side of all the equations of the system have only one argument, $X(N_1, ..., N_k)$ is also not computable. Thus, $Q(N_1, \ldots, N_k) = n(Q^+) - n(Q^-)$, and again, there is a contradiction. We can conclude that if $Q(N_1, \ldots, N_k) \neq n(Q^+) - n(Q^-)$, the system is computable. ■

Conversely to Corollary 5.1, we can also state that:

Corollary 5.2 *Given a PARE, it is undecidable to determine whether there exist values of its parameters, such that the instance of the system is computable.*

Indeed if we invert the occurrences of **constant** and of $X(N_1, ...N_k)$ in the definitions (8), (9), (10), then the new system is computable if and only if $Q(N_1, ..., N_k) = n(Q^+) - n(Q^-)$. The proof is symmetric to the one of Corollary 5.1.

6 Conclusion

This paper has investigated the computability of systems of recurrence equations. We have recalled Joinnault's result which proves that the computability of systems of conditional uniform recurrence equations is undecidable. Moreover, we have shown that the computability of systems of parameterized recurrence equations is also undecidable, by proving that this problem involves the tenth Hilbert's problem.

These results have practical consequences, as far as synthesis of parallel program from recurrence equations is concerned. There are only a few cases when a complete computability analysis can be carried out, as this is possible only for non conditional UREs, or for system of recurrence equations with bounded domain, when the size parameter is fixed, and most of the algorithms found in practice cannot be described using such models. However, the situation is not hopeless, because there exist efficient methods for solving the problem when the computations can be ordered by special orderings such as linear timing-functions. This situation resembles the one which is encountered when writing a program without being able to decide, in the general case, whether it will terminate or not. An open problem is to investigate subclasses of PAREs for which the computability is decidable. However, we could not identify simple restrictions of PAREs that would lead to a better situation, except the well known case of URE.

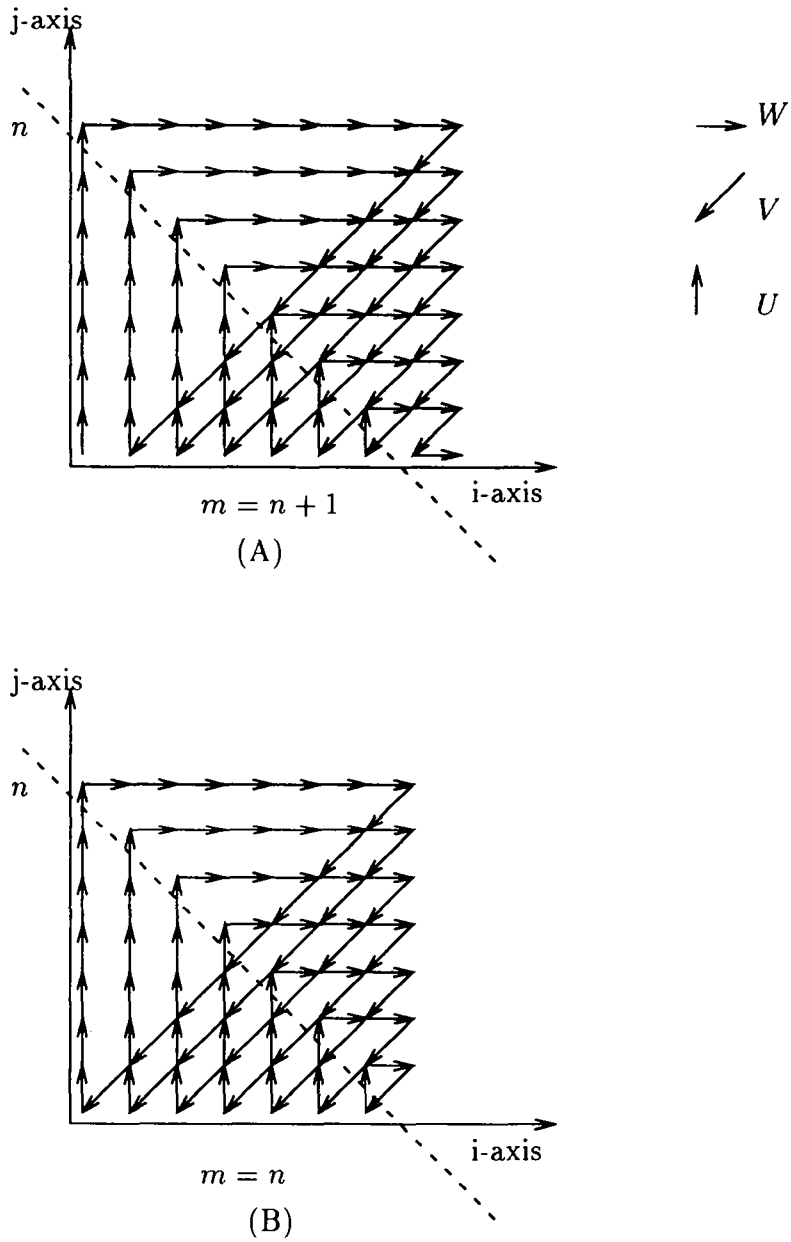


Figure 1: Complete Dependence Graph of Example 2

3.1

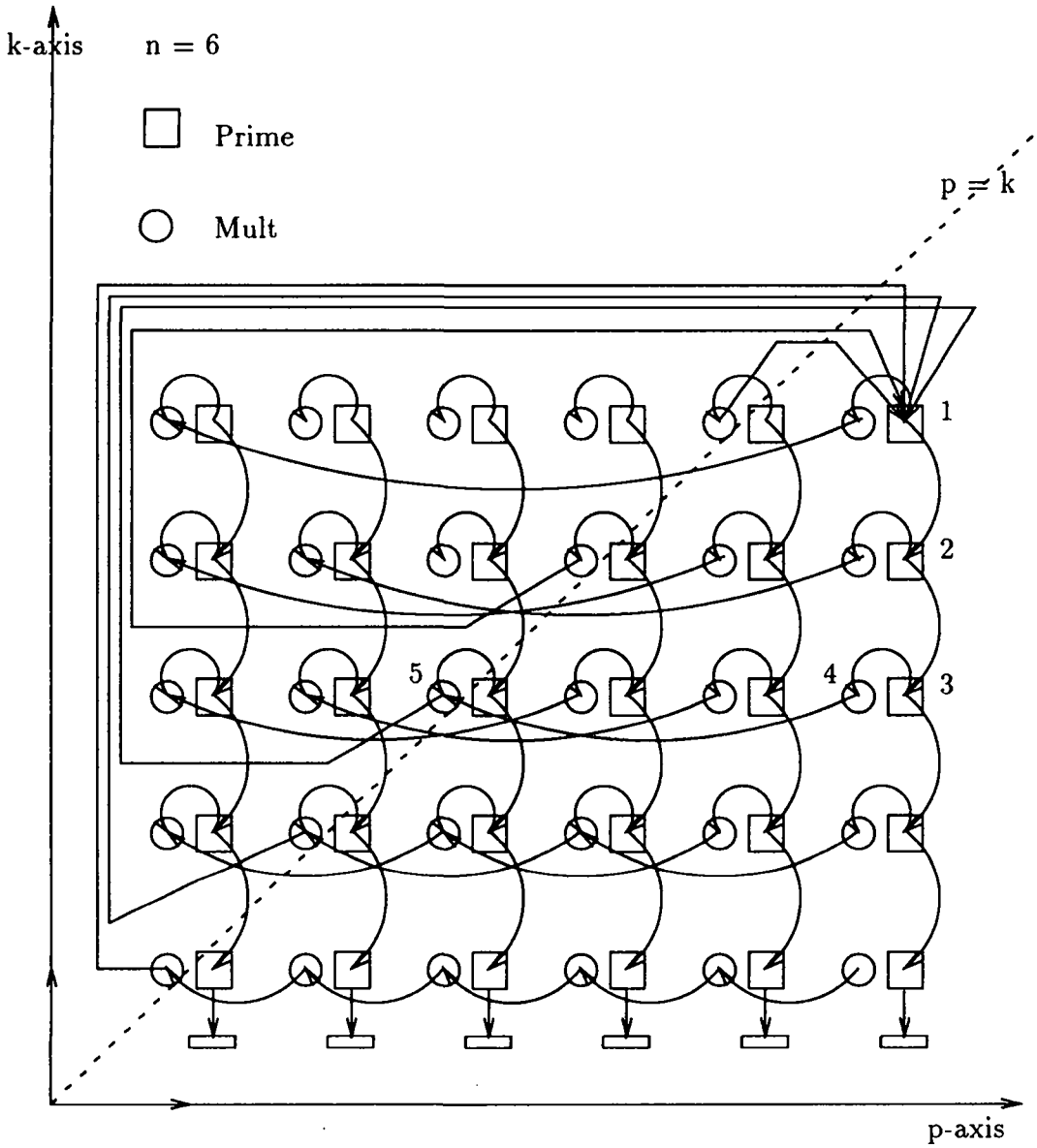


Figure 2: Dependence graph when n is not a prime number : (1,2,3,4,5,1) is a cycle of the dependance graph

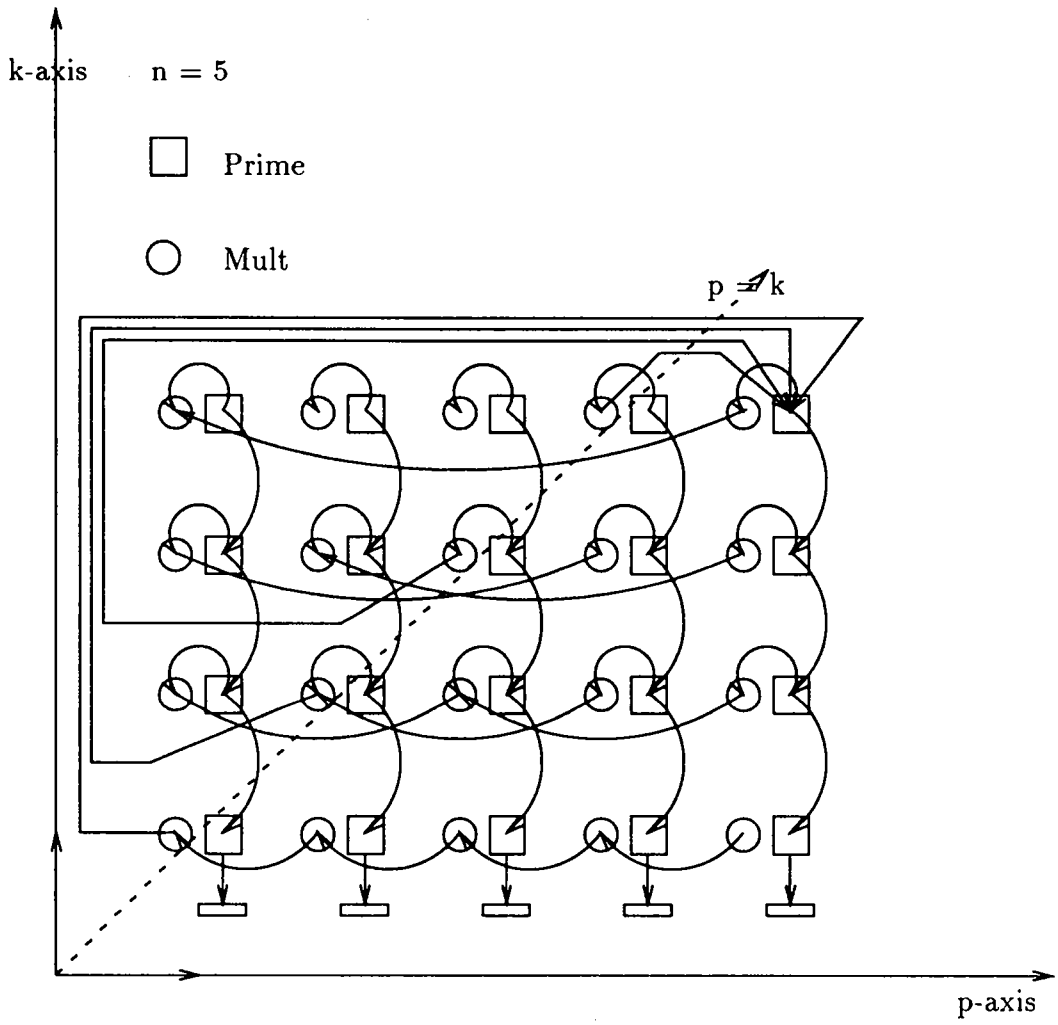
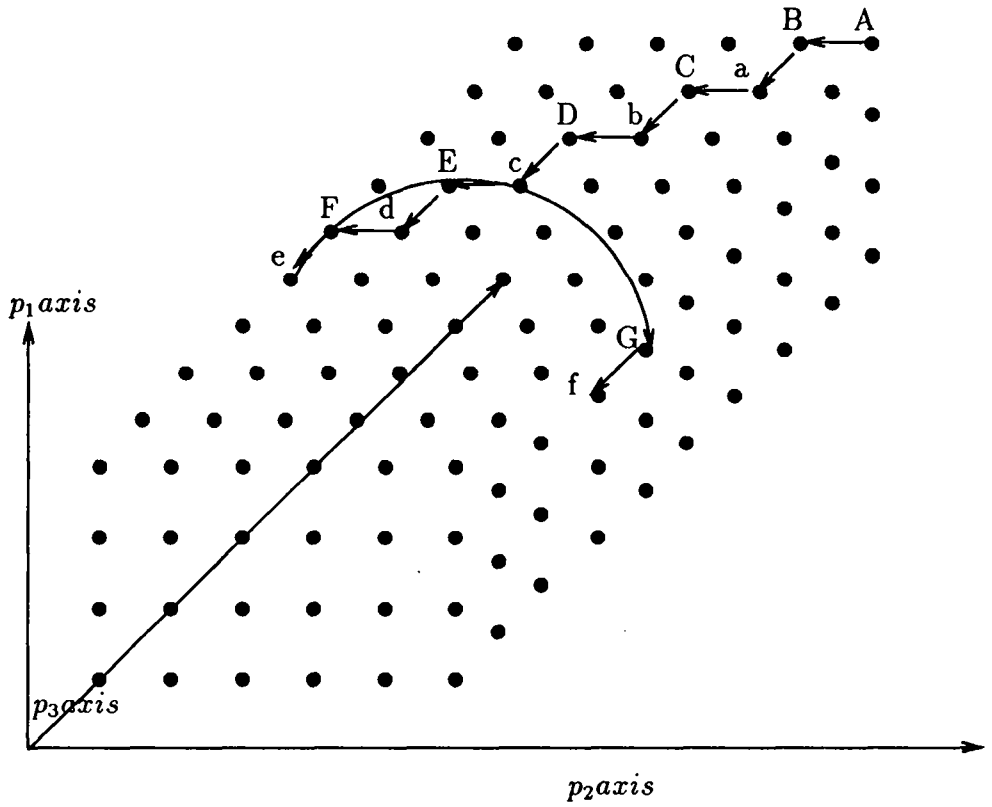
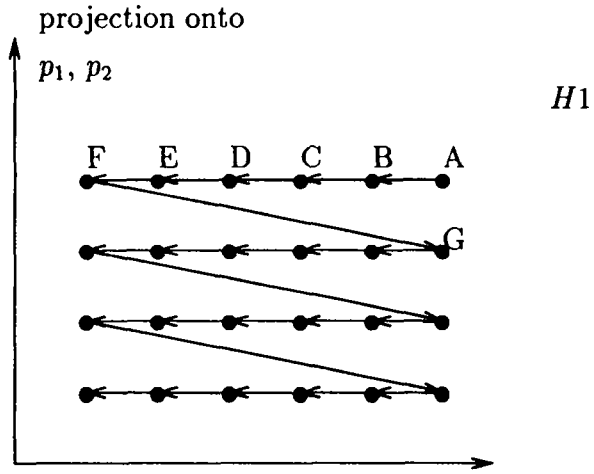
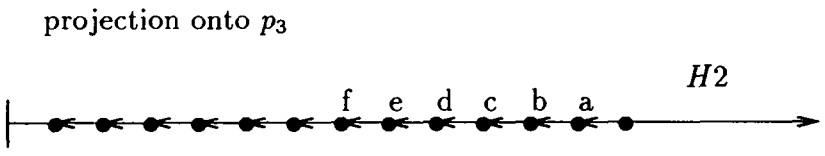


Figure 3: Dependence graph when n is a prime number : There is no cycles.


B
A
C
a
D
b
E
c
F
d
e
G
f
$p_1 axis$
$p_3 axis$
$p_2 axis$


projection onto
p_1, p_2
$H1$
F E D C B A
G


projection onto p_3
$H2$
f e d c b a

References

[1] T. Agerwala and M. Flynn. Comments on capabilities, limitations and correctness of petri nets. In *First Annual Symposium on Computer Architecture*, pages 81–86, 1973.

[2] J.D. Ullman A.V. Aho, J.E. Hopcroft. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, 1974.

[3] J.M. Delosme and I.C.F. Ipsen. Design methodology for systolic arrays. In *Proc. SPIE 30th Ann. Int. Tech. Symp. on Optical and Optoelectronic Applied Sciences and Engineering*, pages 245–59, San Diego (USA), August 1986.

[4] J.M. Delosme and I.C.F. Ipsen. Efficient systolic arrays for the solution of Toeplitz systems: an illustration of a methodology for the construction of systolic architectures for VLSI. In W. Moore, A. McCabe, and R. Urquhart, editors, *International Workshop on Systolic Arrays*, pages 37–46, Adam Hilger, University of Oxford, UK, July 2-4 1986.

[5] D. Hilbert. Gesammelte abhandlungen. *Berlin*, 3, 1935.

[6] B. Joinnault. Conception d'algorithmes et d'architectures systoliques. Thèse de l'Université de Rennes I, Sept 1987.

[7] R.M. Karp, R.E. Miller, and S. Winograd. The organization of computations for uniform recurrence equations. *Journal of the Association for Computing Machinery*, 14(3):563–590, July 1967.

[8] Ju. V. Matijasevic. Enumerable sets are diophantine. *Soviet Mathematics*, 11(2):354–358, March-April 1970.

[9] M.L. Minsky. *Computation : finite and infinite machines*, chapter 6, pages 117–131. Prentice-Hall, Inc. Englewood Clifts, N.J., 1967.

[10] P. Quinton and Y. Robert. *Convolution Systolique de Fonctions Arithmétiques.* Technical Report 449, IRISA, Campus de Beaulieu 35042 Rennes Cédex, janvier 1989.

[11] P. Quinton and V. Van Dongen. The Mapping of Linear Recurrence Equations on Regular Arrays. *The Journal of VLSI Signal Processing*, 1:pages 95-113, 1989.

[12] S.V. Rajopadhye and R.M. Fujimoto. *Synthesizing Systolic Arrays with Control Signals from Recurrence Equations.* Technical Report CIS-TR-86-12, University of Utah, 1987.

[13] S.K. Rao. *Regular Iterative Algorithms and their Implementations on Processor Arrays.* PhD thesis, Standford University, U.S.A., October 1985.

[14] A. Schrijver. *Theory of Linear and Integer Programming. Wiley-Interscience series in Discrete Mathematics*, John Wiley and Sons, 1986.

[15] Y. Wong and J.-M. Delosme. Broadcast removal in systolic algorithms. In K. Bromley, S.Y. Kung, and E. Schwarzlander, editors, *Proc. Int. Conf. on Systolic Arrays*, pages 403–12, IEEE Computer Society Press, 1988.

[16] Yoav Yaacobi and Peter R. Cappello. Scheduling a system of affine recurrence equations onto a systolic array. In *International Conference on Systolic Arrays*, pages 373–382, San Diego (USA), May 1988.

LISTE DES DERNIERES PUBLICATIONS INTERNES IRISA

PI 514 **PARALLELISATION D'UN RESEAU NEURONAL**
Krysztof WOLINSKI
Février 1990, 20 Pages.

PI 516 **COMMENT INTRODUIRE LA CONTIGUITE EN ANALYSE DES CORRESPONDANCES ? Application en segmentation d'image.**
Brigitte ESCOFIER, Habib BENALI, Kaddour BACHAR
Février 1990, 26 Pages.

PI 517 **MACHINE MODELING AND LOOP OPTIMIZATION FOR HORIZONTAL MICROCODED MACHINES**
François BODIN, François CHAROT
Février 1990, 24 Pages.

PI 518 **MULTISCALE SYSTEM THEORY**
Albert BENVENISTE, Ramine Nikoukhah, Alan S. Willsky.
Février 1990, 30 Pages.

PI 519 **PANDORE : A SYSTEM TO MANAGE DATA DISTRIBUTION**
Françoise ANDRE, Jean-Louis PAZAT, Henry THOMAS
Février 1990, 14 Pages.

PI 520 **SCHEDULING AFFINE PARAMETERIZED RECURRENCES BY MEANS OF VARIABLE DEPENDENT TIMING FUNCTIONS**
Christophe MAURAS, Patrice QUINTON
Sanjav RAJOPADHYE, Yannick SAOUTER
Février 1990, 14 Pages.

PI 521 **COMPUTABILITY OF RECURRENCE EQUATIONS**
Yannick SAOUTER, Patrice QUINTON
Février 1990, 28 Pages.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399