1 Fourier's Elimination: Which to Choose ?

Jean-Louis Imbert

1.1 Abstract

This paper is concerned with variable elimination methods in linear inequation systems, related to Fourier's elimination [8]. Our aim is to make visible the links between the different contributions of S.N. Černikov [3], D.A. Kolher [17], R.J. Duffin [6], JL.J. Imbert [10], and J.Jaffar, M.J. Maher, P.J. Stuckey and R.H.C. Yap [14]. We show that the three methods proposed by Černikov, Kolher and Imbert produce exactly the same output (without more or less redundant inequations), up to multiplying by a non-zero positive scalar. We present and discuss the improvements of Černikov, Duffin, Imbert and Jaffar et al., and propose a new improvement. We give a short analysis of the complexity of the main improvements and discuss the choice of the method in relation to the problem at hand. We propose a pattern algorithm. Finally, we conclude with a comparative assessment through a brief example and a few remarks.

1.2 Introduction

Variable elimination is of major interest for Constraint Logic Programming Languages [12], and Constraint Query Languages [15], where we would like to eliminate auxiliary variables introduced during the execution of a program. This elimination is always suitable for final results. It can also increase the efficiency of the intermediary processes. We focus on linear inequalities of the form $ax \leq b$, where a denotes a n-real vector, x an n-vector of variables, b a real number, and the juxtaposition ax denotes the inner product. This type of constraint occurs in CLP languages such as CHIP [7, 22], CLP($\Re$) [13], and Prolog III [4, 5]. A constraint system is a conjunction of constraints. Using matrix notation, an inequation system $\{a_i x \leq b_i \mid i = 1, \ldots, m\}$ can be written $Ax \leq b$, where A denotes an $(m \times n)$-matrix, and b an m-real vector. The main problem we face during the variable elimination process in linear inequation systems, is the size of the output. It is doubly exponential. Variable elimination in inequation systems has been extensively investigated. Among these investigations one can cite the C. and JL. Lassez

method [18], which globally eliminates in one single operation the set of unwanted variables. It is based on semantic properties of projection and of convex hull. It makes it possible to obtain approximations. Nevertheless, so far, mainly methods derived from Fourier's elimination [8] are used in CLP languages. For example, the programming language CLP($\Re$) outputs its results after eliminating undesirable variables using a method related to Fourier [14]. This kind of method carries out an incremental elimination of variables, one after another. The major problem comes from the size of intermediary systems.

In this paper, our interest will be exclusively focused on methods derived from Fourier's elimination. Among the improvements in these eliminations are the contributions of S.N. Černikov [3], D.A. Kolher [17], R.J. Duffin [6], JL.J. Imbert [10], and J.Jaffar, M.J. Maher, P.J. Stuckey and R.H.C. Yap [14]. There are basically three approaches represented in these methods: the general algebraic approach of Černikov, the matricial algebraic approach of Kohler, and the graph, or more specifically, the tree approach of [10]. These trees reflect the way in which the new inequations are constructed as the elimination proceeds. This last approach shows clearly that the methods proposed by Černikov, Kohler and Imbert are equivalent. This is far from being obvious as Kohler remarks (see Section 1.5).

The aim of this paper is to make visible the links between these different above-mentioned contributions. This study, which has not been done before, is of great interest. We show that the three methods proposed by Černikov, Kolher and Imbert produce exactly the same output (without more or less redundant inequations), up to multiplying by a non-zero positive scalar. We present and discuss improvements by Černikov, Duffin, Imbert and Jaffar et al., and propose a new improvement. We give a short analysis of the complexity of the main improvements and discuss the choice of the method depending on the problem at hand. We propose a pattern algorithm. Finally, we conclude with a comparative assessment using a brief example and a few remarks.

The rest of this paper is organized in the following way: Section 1.3 presents basic concepts necessary for understanding the other Sections. In Section 1.4, we introduce the parts of the initial system (called minimal parts) which can produce a relevant final inequation, and present the Černikov-Fourier algorithm [3]. In Section 1.5, we introduce the characterization of minimal parts and give the Kohler-Fourier algorithm

[17]. Moreover, in this Section, we show that the algorithms of Černikov-Fourier and Kohler-Fourier produce exactly the same final system (without more or less redundant inequations), up to multiplying each inequation by a non-zero positive scalar. Section 1.6 successively presents improvements by Černikov, Duffin, Imbert and Jaffar et al., and introduces some new precisions about redundancy. Furthermore, we show (subsection 1.6.4) a new improvement of the Černikov minimal part method, dividing in half the average number of comparisons. In Section 1.7, we discuss the complexity of the various improvement contributions and the choice between the minimal part method (Černikov) and the matricial method (Kohler). Then we give a pattern variable elimination algorithm for inequation systems. Section 1.8 compares, through a brief example, the contributions of Černikov, Kohler and Imbert [10]. Finally, in Section 1.9, we conclude with a few remarks.

1.3 Preliminaries

Let x denote the vector $(x_1, \ldots, x_n)$. Let x' be the vector of variables to be retained, and let x'' be the vector of variables to be eliminated. We will abuse the language by writing $x = (x', x'')$. In the same way, $a = (a_1, \ldots, a_n)$ denotes a real or rational vector, and a will be written (a', a''), respective of the subscripts of x' and x''. The inequation system $Ax \leq b$ can then be written $A'x' + A''x'' \leq b$. To eliminate the variables of x'', the problem at hand is to find a system $Cx' \leq d$ as concise as possible for which, if (a', a'') is in the solution set of $A'x'+A''x'' \leq b$, then a' is in the solution set of $Cx' \leq d$, and reciprocally, if a' is a solution of system $Cx' \leq d$, then there is a'' such that (a', a'') is a solution of system $A'x' + A''x'' \leq b$. We will say that $Cx' \leq d$ and $A'x' + A''x'' \leq b$ are *equivalent on* x'. It can be shown that each inequation of the final system $Cx' \leq d$ is a linear combinatory with positive coefficients of inequations of initial system (this is a consequence of Fourier's elimination).

1.3.1 Fourier's Algorithm

Let $a_1 x \leq b_1$ and $a_2 x \leq b_2$ be two inequations, and let $a_{1,1}$ and $a_{2,1}$ be the coefficients of the variable x_1 respectively in the first and second inequations. Let us assume that $a_{1,1} > 0$ and $a_{2,1} < 0$. Then

$$-a_{2,1}(a_1 x) + a_{1,1}(a_2 x) \leq -a_{2,1} b_1 + a_{1,1} b_2 \qquad (1.3.1)$$

is a consequence of the two initial inequations, and x_1 does not occur in it. Let $Ax \leq b$ be an inequation system, and V the variables which occur in it. Let $\widetilde{A}x \leq \widetilde{b}$ be the system obtained from $Ax \leq b$, by removing all inequations which x_1 occurs in, and replacing them with all the inequations of type (1.3.1) above, we can construct from any pair of removed inequations. It can be proven that $\widetilde{A}x \leq \widetilde{b}$ and $Ax \leq b$ are equivalent on $V - \{x_1\}$. This operation must be successively repeated for each variable to be eliminated.

1.3.2 Redundancy and size of the intermediary systems

A drawback of Fourier's elimination is the production of redundancies which overwhelms the system. It is here that the improvements of S.N. Černikov [3], D.A. Kolher [17], R.J. Duffin [6], JL.J. Imbert [10], and J.Jaffar, M.J. Maher, P.J. Stuckey and R.H.C. Yap [14], come. One way to mitigate this drawback is to associate each inequation with information to memorize the way in which it has been produced. This information deals with: the inequations of the initial system used to produce this inequation, the variables which are effectively eliminated during pair gathering, and, finally, the other variables eliminated during the previous variable eliminations. Some relations between the quantities of elements occuring in these three kinds of variables, make it possible to detect whether a new inequation is redundant or not.

Another drawback of Fourier's method is the large increase in the number of constraints in the intermediary systems. One of the main reasons is that the historic method quickly detects most of the redundancies, but it does not detect all of them (only those due to the construction, i.e. due to $A''x''$, but not those due to $A'x'$). Besides, redundancies have a tendency to spread because they multiply very quickly. The other reason concerns the nature of Fourier's method: independently of the redundancies, the number of inequations has a tendency to increase, before decreasing when few variables remain to be eliminated. A remedy for the first cause is to use a general method of redundancy removing [20, 16, 19, 11]. As for the second cause, which is structural, it cannot be suppressed. Other methods are necessary for remedying this inconvenience. Numerous methods have been proposed, including one already cited and set out in [18] which is particularly interesting as it creates a minimal representation of the final system and does not use intermediary systems.

1.3.3 Some results

Let S be the linear inequation system $\{a_1x \leq b_1, \ldots, a_nx \leq b_n\}$. It is known (Farkas [21, p87-90]), that the inequation $cx \leq d$ is a consequence of S if and only if there exist positive coefficients (≥ 0) $\alpha_0, \alpha_1, \ldots, \alpha_n$, such that $c = \alpha_1 a_1 + \ldots + \alpha_n a_n$ and $d = \alpha_0 + \alpha_1 b_1 + \ldots + \alpha_n b_n$. The inequation $cx \leq d$ is said to be an *affine combinatory with positive coefficients* of inequations of S. A *linear combinatory* is an affine combinatory such that $\alpha_0 = 0$. Hence, an inequation of S is redundant in S if and only if it is an affine combinatory with positive coefficients of the other inequations of S. It is *weakly redundant* if $\alpha_0 = 0$ in each affine combinatory, *strongly redundant* in other cases.

Lastly, let us notice that, from Fourier's elimination, each inequation obtained after variable elimination in an inequation system S, is a linear combinatory with positive coefficients of the inequations of S.

1.4 Minimal subset

As a result of Fourier's elimination (the last remark of Section 1.3.3), each inequation of the final system $Cx' \leq d$ is a linear combinatory with non-negative coefficients of inequations of the initial system $A'x' + A''x'' \leq b$. Hence, we look at the real-vectors $w = (w_1, \ldots, w_m)$ such that $wA'' = 0''$. ($0''$ denotes the vector a'' with all its components equal to zero. The vector a'' has been introduced in Section 1.3. We will abuse the language by writing 0 instead of $0''$ if there is no ambiguity. In the same way, we will write 0 instead of $0'$). A vector w_1 is less than or equal to a vector w_2 (written $w_1 \leq w_2$), if each component of w_1 is less than or equal to its corresponding component of w_2. Let F be the cone[1] of non-negative solutions $0 \leq w$ of $wA'' = 0$. From the last remark of Section 1.3.3, it is evident that:

LEMMA 1 The system $\{wA'x' \leq wb \mid w \in F\}$ is equivalent to the final system $Cx' \leq d$.

However, this system is unnecessarily large. Indeed, on the one hand, if w_1 and w_2 are two elements of F identical up to multiplying by a non-zero scalar, then the inequations $w_1A'x' \leq w_1b$ and $w_2A'x' \leq w_2b$ are

[1] a cone is a point set containing all the linear combinatories with positive coefficents of any finite number of its elements.

equivalent. On the other hand, from [1, Section 3 p 297-298] it suffices that the number of non-zero components of w is less than or equal to the number of non-zero components of x'' plus 1. As a matter of fact, according to [3], it is sufficient to take a base of F. A *base* of F is an irreducible subset of elements of F which generates F. Here, irreducible is for inclusion. Since F is a cone, a vector is *generated* by other vectors if it is a linear combinatory with non-negative coefficients of these other vectors.

Two elements of F are *essentially different* if they do not differ from one another by a positive scalar multiple. A *minimal vector* is a non-zero vector of F such that the only essentially different vector of F less than itself, is the zero-vector. Let G be a base of essentially different minimal vectors of F. Then, using lemma 1, it is evident that:

THEOREM 1 The systems $Cx' \leq d$ and $\{wA'x' \leq wb \mid w \in G\}$ are equivalent.

Moreover,

COROLLARY 1 If w_1 and w_2 are two elements of G, the zero-components of which coincide with each other, then $w_1 = w_2$.

Proof 1 If $w_1 \neq w_2$, there is a linear combinatory of these two vectors which produces a non-zero element of F less than each of them. Indeed, it can be found $k > 0$, such that $kw_1 \leq w_2$. Then, continuously increase k with $kw_1 \leq w_2$, until at least one non-zero component of kw_1 is equal to the corresponding component of w_2. Then, $w_2 - kw_1$ is non-zero positive and is in F. So, w_1 and w_2 are not minimal and are not in G.

Each minimal vector of F or G, is associated with an inequation subset of $A'x' + A''x'' \leq b$ (the lines of the initial subset corresponding to the non-zero components of the minimal vector). This subset is said to be *minimal for* x'', or simply *minimal* if there is no ambiguity.

From [1, section 3 p 297-298] and [3], theorem 1 can be used at each step in Fourier's elimination, yielding a correct algorithm. Then, for each inequation i, we memorize the subset H_i of initial inequations (i.e., of initial system $Ax \leq b$) which produced i. H_i is called a *historical subset* (a *label* in [14]). Clearly, for each initial inequation i, $H_i = \{i\}$. In Fourier's algorithm, only inequations with minimal historical subsets are retained. An algorithm using these properties is given in Figure 1.1.

Input: $Ax \leq b$, a system of inequations.
Assume that $x = (x', x'')$, where $x'' = (x_1, \ldots, x_k)$ is the vector of variables to be eliminated.
Output: A system $Cx' \leq d$ equivalent to $Ax \leq b$ on x'

begin

1. At the start, the inequation system S is equal to $Ax \leq b$.
2. **For** x_j being successively the variables $x_1, \ldots, x_k$,
   - 2.1. Remove from S, each inequation with a non-zero coefficient of x_j.
   - 2.2. For each pair of removed inequations $a_1 x \leq b_1$ and $a_2 x \leq b_2$, with components $a_{j,1}$ and $a_{j,2}$ of x_j respectively positive and negative, insert the inequation $a_{j,1}a_2 x - a_{j,2}a_1 x \leq a_{j,1}b_2 - a_{j,2}b_1$ in S. If the historical subsets of the inequations of the pair are H_1 and H_2 respectively, then, the historical subset of the new inequation is $H_1 \bigcup H_2$.
   - 2.3. Remove from S each inequation with a historical subset including at least one historical subset of the remaining inequations.
3. Return the system S which is then of the form $Cx' \leq d$.

end

Figure 1.1
Černikov-Fourier's Algorithm

Notice that the historical subset of the new inequation, and the comparison can be processed before producing the inequation. Hence, the inequation is not created if at least one existing historical subset is included in its historical subset.

Example 1 Let us consider the following system:

$$\begin{array}{ll} (1) & x_1 - x_4 - x_5 \leq 2 \\ (2) & x_2 - 2x_4 + x_5 \leq 0 \\ (3) & -x_1 + x_2 + x_4 - x_5 \leq 0 \\ (4) & -x_3 + x_4 + 2x_5 \leq 0 \end{array}$$

Assume that $x' = (x_1, x_2, x_3)$ and $x'' = (x_4, x_5)$. This system can be written in matricial form:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} -1 & -1 \\ -2 & 1 \\ 1 & -1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The vectors $w_1 = (3, 0, 1, 2)$, $w_2 = (5, 2, 5, 4)$, $w_3 = (1, 1, 2, 1)$ and $w_4 = (0, 3, 5, 1)$ are in F. The vector w_1 is minimal in F because each non-zero vector w of F in which the second component is 0, is of the form $(3k, 0, k, 2k)$ where k is a real. The corresponding minimal part is formed of the three inequations (1), (3) and (4). The vectors w_2 and w_3 are not minimal since the vector $1/3\ w_1$ is less than and essentially different from each of them. And for any non-negative real k, the vector $k\ w_1$ is in F.

Moreover, $G = \{w_1, w_4\}$ is a base of F, since $w_2 = 5/3\ w_1 + 2/3\ w_4$ and $w_3 = 1/3\ w_1 + 1/3\ w_4$.

1.5 Characterization of minimal subsets

The main problem we face in the previous method is that the historical subset of each inequation of $Cx' \leq d$ must be compared to the historical subset of every other inequation of $Cx' \leq d$. The following theorem makes it possible to overcome this drawback. In the following, the notion of rank of vector space, affine space, vector system and matrix is assumed known.

THEOREM 2 Let $A'x' + A''x'' \leq b$ be an inequation system. Let F be the cone defined in Section 1.4. An element w of F is minimal iff the sub-matrix A''_w formed from lines of A'' related to non-zero components of w has as its rank, the number of lines of A''_w minus 1.

Proof 2 Since $wA'' = 0$, this rank is at most the number of lines of A''_w minus 1. If it is smaller, there exists a vector v of F with at least one non-zero positive component, such that for each non-zero component of v, the corresponding component of w is non-zero, and such that $vA'' = 0$. Without loss of generality, v can be assumed less than or equal to w, with equality for at least one non-zero component (if necessary, multiply by an appropriate scalar). As a result, $w - v$ is a non-zero vector of F, then w is not minimal.

Example 2 Let us take up again the previous example. Then w_1 is minimal since the matrix

$$A''_{w_1} = \begin{pmatrix} -1 & -1 \\ 1 & -1 \\ 1 & 2 \end{pmatrix},$$

evidently has rank 2 with three rows.

But, w_2 is not a minimal vector, since the matrix

$$A''_{w_2} = \begin{pmatrix} -1 & -1 \\ -2 & 1 \\ 1 & -1 \\ 1 & 2 \end{pmatrix},$$

has rank 2 and four rows.

In fact, Černikov defines a minimal vector (which he calls *fundamental element*) as a vector w for which the rank of the matrix A''_w is the number of its lines minus 1 [3, p 1520]. He then says that the maximal systems of essentially different minimal elements of F are identical with its bases. Kohler used this result in Fourier's elimination algorithm. In the algorithm of Figure 1.2, if H is the historical subset associated with an inequation, A''_H will denote the matrix formed from lines and columns of A'' which respectively correspond to elements of H and to columns of $x_1, \ldots, x_j$.

Note that, contrary to the Černikov-Fourier Algorithm, the detection of minimal subsets is performed at the time of the creation of the new inequations. That is to say, whether a subset is minimal or not is detected at the time of its creation. This is the opposite of the Černikov-Fourier Algorithm, in which some inequations are temporarily kept until a new inequation with a smaller historical subset is created. As a result, time used for creating non-retained inequations, and an non-negligible place used for intermediary storage can be saved.

However, it can be asked whether this method detects when the same inequation can be obtained from the same minimal subset in more than one way. Kohler [17, p 23]: *"I have been unable to prove that we can discount the possibility of the Fourier-Motzkin Method generating more than one extreme vector from the same half-line. Should this occur we need only keep one of them"*. The tree approach of [10] makes it possible

Input: $Ax \leq b$, an inequation system.
Assume that $x = (x', x'')$, where $x'' = (x_1, \ldots, x_k)$ is the vector of variables to be eliminated.
Output: A system $Cx' \leq d$ equivalent to $Ax \leq b$ on x'

begin

1. we begin with the system of inequations S equal to $Ax \leq b$.
2. **For** x_j being successively the variables $x_1, \ldots, x_k$,
   - 2.1. Remove from S, each inequation with a non-zero coefficient of x_j.
   - 2.2. **For each** pair of removed inequations $a_1 x \leq b_1$ and $a_2 x \leq b_2$, with components $a_{j,1}$ and $a_{j,2}$ of x_j respectively positive and negative,
     - 2.2.1 If the historical subsets of the inequations of the pair are H_1 and H_2 respectively, then, the historical subset of the new inequation is $H_1 \bigcup H_2$.
     - 2.2.2 If the rank of A''_H is its number of lines minus 1, insert in S the new inequation $a_{j,1} a_2 x - a_{j,2} a_1 x \leq a_{j,1} b_2 - a_{j,2} b_1$.
3. Return the system S which is then of the form $Cx' \leq d$.

end

Figure 1.2
Kohler-Fourier's Algorithm

to give a simple answer to this question using its unicity theorem (p 121). This theorem can be translated as follows:

THEOREM 3 (UNICITY THEOREM) For each minimal subset, Fourier's algorithm, modified by Černikov or Kohler, can produce only one inequation and only in one way.

COROLLARY 2 The Černikov-Fourier and Kohler-Fourier algorithms rigorously produce the same final system without more or less redundant inequation.

Proof 3 It is an immediate consequence of the previous theorem, since, as a result of Theorem 2, the Kohler-Fourier algorithm produces all the inequations produced by the Černikov-Fourier algorithm, and eventually, some double inequations.

Remark : Theorem 3 is valid only if, at each Fourier step, each inequation associated with a non-minimal historical subset is discarded. Otherwise uniqueness is not guaranteed and the Černikov method is preferred. In addition, it can be remarked that this theorem involves the result of corollary 1.

Another very interesting result shown in [10] is the independence of the order in which the variables are eliminated:

THEOREM 4 Whatever the order in which the variables are eliminated, the Černikov-Fourier algorithm or the Kohler-Fourier algorithm rigorously produce the same final system (up to multiplied by a positive scalar), without more or less redundant inequation.

1.6 Quick detection of minimal or non-minimal subsets

The main problem we face in the previous two methods is that the minimal subset detection operation is very costly. In this section, we present various known solutions to this problem and we give some new answers. Most of these improvements can be applied independently to both methods.

1.6.1 Upper limit of the rank

The first improvement was provided by the precursor of the previous two methods. Though Černikov presents the following improvement in [3], we can find its foundation in [1, p 296 corollary 2].

THEOREM 5 After k variable eliminations, if the historical subset associated with an inequation has more than $k+1$ elements, then this historical subset is not minimal.

Proof 4 It can be immediately deduced from theorem 2, and from the fact that the matrix A''_w has k columns, and then its rank is less than or equal to k.

In this case, the detection cost is very low. Černikov and Kohler included this detection in their algorithms.

1.6.2 Passive variables

Duffin, in [6, p 90], introduces the active or passive variable concept. A variable is *active* during its elimination if there is at least one pair of inequations in the sense of Fourier's elimination. Otherwise the variable is said to be *passive*. Let x_j be a passive variable. Its elimination can suppress some inequations from the system, but does not add any. This comes from the fact that the coefficients of x_j in the inequations are either all positive ($0 \leq$) or all negative (≤ 0). Then if we delay the elimination of such a variable, since Fourier's elimination uses only linear combinatories with positive coefficents, the coefficients of x_j will all have the same sign in all generated inequations. In particular, the inequations with non-zero coefficients of x_j, generate inequations with non-zero coefficients of x_j. These inequations will all be rejected in a subsequent elimination of this variable. Thus,

THEOREM 6 After the elimination of k variables, p of which are passive, if in a historical subset more than $k + 1 - p$ elements occur, then this historical subset is not minimal.

1.6.3 Upper and lower limits of the rank

So far, we have looked at the improvements at a global level. However, the minimal subset is a local concept in that a minimal part depends only on what is included within it. Hence, it is sufficient to look at the eliminated variables occuring in at least one inequation of the historical subset [10].

Assume that the variables eliminated from the initial system are $x_1, \ldots, x_k$. We will say that they are *officially eliminated*, and will write O_k the set of these variables. For each inequation i produced, the set O_k can be divided into three disjointed subsets: the subset of *effectively eliminated* variables denoted E_i, the subset of *implicitly eliminated* variables denoted I_i, and the other variables. A variable is said to be effectively eliminated for i, if its official elimination produces at least one of its ancestors (initial or intermediate inequations used to produce i). A variable is said to be implicitly eliminated for i, if the following three conditions are satisfied: it occurs in at least one inequation of H_i, it does not occur in i, it is not effectively eliminated for i. In practice, in I_i we also find some implicitly eliminated variables which are not in O_k.

Example 3 Let us consider the following system from which we want to eliminate the variables x_1, x_2, x_3 and x_4:

$$\begin{array}{ll} (1) & x_1 - 2x_3 - x_4 + x_6 + x_7 \leq 2 \\ (2) & -x_1 + 2x_3 - x_5 - x_6 \leq 1 \\ (3) & x_1 + x_3 + x_4 + 2x_7 \leq 0 \\ (4) & -x_1 - x_3 \leq -1 \end{array}$$

Among the inequations produced by the elimination of x_1 are the two following ($(5) = (1) + (2)$ and $(6) = (3) + (4)$).

$$\begin{array}{ll} (5) & -x_4 - x_5 + x_7 \leq 3 \\ (6) & +x_4 + 2x_7 \leq -1 \end{array}$$

The variables x_2 and x_3 do not occur in the inequations (5) and (6). The elimination of these two variables retains these two inequations. To eliminate x_4, we only have to add (5) and (6).

$$(7) \quad -x_5 + 3x_7 \leq 2$$

The officially eliminated variables are x_1, x_2, x_3 and x_4. The variables effectively eliminated for the inequation (7) are x_1 and x_4. The variable x_3 has been implicitly eliminated for the inequation (7), as has been the variable x_6 which is not officially eliminated. The variable x_2 is eliminated neither effectively nor implicitly for (7), since it does not occur in any of the four original inequations used to produce (7).

In [10] the following two theorems are shown, where $Card(S)$ denotes the number of elements of S:

THEOREM 7 (FIRST ACCELERATION THEOREM) If H_i is a minimal subset, then the following relation is satisfied:

$$1 + Card(E_i) \quad \leq \quad Card(H_i) \quad \leq \quad 1 + Card(E_i \bigcup (I_i \bigcap O_k))$$

In this same paper it is shown that whatever H_i, minimal part or not, the left inequality is satisfied. The right inequality gives an upper limit less than or equal to the one of Theorem 5 above. In fact, from Theorem 2, Theorem 5 says that the rank of the A''_w matrix can be more than the number of columns. Theorem 7 suppresses from this number the zero columns. Moreover, the Duffin improvement is always global

and still applicable: we only have to suppress passive variables from O_k. When the first acceleration theorem quickly detects non-minimal parts, the second acceleration theorem quickly detects minimal parts:

THEOREM 8 (SECOND ACCELERATION THEOREM) Let i be an inequation such that $1 + Card(E_i) = Card(H_i)$, then H_i is minimal.

This theorem avoids a heavy verification burden. The cost of these two theorems is very low. It linearly depends on the number of eliminated variables. The costly research operation of minimal subsets, either by comparison of minimal parts, or by computing a matrix rank, needs to be done only when

$$1 + Card(E_i) < Card(H_i) \leq 1 + Card(E_i \bigcup (I_i \bigcap O_k)).$$

Thus, if there is no implicitly eliminated variable, these two theorems are sufficient. Usually, this happens when the constraints are randomly generated. In return, the algorithm performances decrease when the number of implicitly eliminated variables increases.

An example using these two theorems is given in section 1.8.

1.6.4 Comparison number

The following improvement deals only with the Černikov-Fourier method. The set G is a base of F, and there is a one-to-one map between G and the set of minimal subsets. Then, for each element j of a non-minimal subset P, there is a minimal part included in P, in which j occurs. As a result, if we take an ordering on the initial inequations, the comparison between subsets with the same first element is sufficient. So, the average number of comparisons is divided in half. However, more storage space is needed for intermediary results.

Example 4 Let us take up again Example 1. The vector w_2 produces the inequation $7x_2 - 4x_3 \leq 10$ associated with the historical set $\{(1), (2), (3), (4)\}$. This historical set is not minimal. Then it includes a minimal historial set in which occurs the inequation (1). Indeed, the vector w_1 produces the inequation $2x_1 + x_2 - 2x_3 \leq 6$ associated with the historical set $\{(1), (3), (4)\}$. In fact, if the original system is without redundancy, it can be shown that $H_{w_2} = H_{w_1} \bigcup H_{w_4}$, since $w_2 = 5/3\ w_1 + 2/3\ w_4$ and since the vector w_4 produces the inequation $-5x_1 + 8x_2 - x_3 \leq 0$ associated with the historical set $\{(2), (3), (4)\}$.

1.6.5 Redundancies

Let us consider the initial inequation system $A'x' + A''x'' \leq b$. So far, the only redundancies suppressed during the elimination process of x'', are the ones due to A''. However, if we take into account $A'x'$, other redundancies may appear. I do not know a method which suppresses all redundancies, compatible with one of Fourier elimination methods presented above. However, in some cases, coexistence is possible. In [14], it is shown that strong redundancies[2] produced by Fourier's algorithm with the previous improvements, can be suppressed without subsequent damage.

THEOREM 9 Every inequation, at least one ancestor of which is strongly redundant, is strongly redundant.

Proof 5 Let $ux \leq v$ be a strongly redundant inequation. Let us assume that this inequation is equal to the linear combinatory $(\sum_{i=1}^{i=p} \alpha_i a_i)x \leq (\sum_{i=1}^{i=p} \alpha_i b_i) + \alpha_0$. The proof is then trivial since $uv \leq v - \alpha_0$ is a logical consequence of the inequation system and since each produced inequation is a linear combinatory with positive coefficients of the inequations of the system.

The systematic detection of strong redundancies is very costly. However, there are some cases in which the detection cost is lower. The quasi-redundancy is a special case of strong redundancy. An inequation $ux \leq v$ is quasi-redundant in $Ax \leq b$ if there is another inequation of that system written $ux \leq v - r$ up to multiplied by a positive scalar, where r is a non-zero positive constant [19]. The quasi-redundancy detection is not excessively expensive because it is roughly a one-to-one comparison of constraints with each other.

Remark : Assume that all or part of the strong redundancies are suppressed from an intermediary system $\mathcal{C}_i$, and that $\mathcal{C}_i$ is obtained by a derived Fourier's elimination method $\mathcal{M}$. Let $\mathcal{K}_i$ be the subsystem of $\mathcal{C}_i$ so obtained. To take advantage of this, we have to be sure that in the next steps all the redundancies detected by $\mathcal{M}$ applied on $\mathcal{C}_i$, will not occur in $\mathcal{K}_{i+1}$ when this same method is applied on $\mathcal{K}_i$. If this is

[2] the supporting hyperplane of which is far from the solution set of the inequation system.

the case, we will say that the method $\mathcal{M}$ is *fully compatible* with the partial or full deletion of strong redundancies. Otherwise we have to find a means to detect all strong redundancies at each step, and this detection is too costly, and thus impracticable.

The methods of Černikov-Fourier and Kohler-Fourier, are opposed in that the first detects the minimal subset by comparison with each of the others, whereas the second only needs to know the inequations of its historical subset. As a result, if some minimal subsets are missing due to strong redundancy deletions, the comparison method can be put on the wrong track. On the contrary, this is not the case for the matricial method. In conclusion we have the following result

PROPOSITION 1 Any partial deletion of strong redundancies is fully compatible with the Kohler-Fourier elimination method but is not fully compatible with the Černikov-Fourier elimination method.

Thus, the partial deletion of strong redundancies is not suitable for the Černikov-Fourier elimination method.

1.7 Comparison or matricial computation ?

1.7.1 Complexity

The incorporation of improvements from theorems 5, 6, 7 and 8, results in a great reduction in the cost of the elimination algorithm. If m_0 is the number of inequations of the initial system, and if k is the number of variables to be eliminated, the cost of theorems 5, 7 and 8 is at most $O(m_0 + k)$ for each produced inequation. Note that the detection of the minimality of historical subsets can always be done before the creation of the new inequations. Thus, time can be saved in cases of rejection.

The cost of minimal subset detection using comparison is, at most, $O(m_0 m)$ for each produced inequation where m is the maximal number of inequations occurring in the intermediary system during the process of elimination. It must be noted that in the case of the Černikov-Fourier method, the comparison must be done in both directions of the inclusion. If we use the improvements of theorems 7 and 8, the comparison must be done only for inequations satisfying theorem 7: in both directions of the inclusion between historical subsets of two new inequations which do not satisfy theorem 8, only in one direction when one of the new

inequations satisfies theorem 8, no comparison is needed when the two inequations satisfy this same theorem.

The cost of minimal subset detection using matricial computation is, for each produced inequation, at most $O(k^5)$ in infinite precision, $O(k^3)$ otherwise. Thus, when k is low, it is better to use matricial computation and to change methods during the elimination process when k and m move. Theorem 3 and its corollary allow for this change at every moment.

Furthermore, in the choice of method, we have to take into account the fact that the matricial method allows for an independent process of produced inequations. Conversely, if during the elimination of a variable, the comparison is used, there will be comparisons to do until the end of the elimination of that variable. Moreover, the comparison method does not always immediately detect redundancies, as a result, this leads to an additional cost because of the intermediary storage. Consequently, the matricial method allows for a degree of parallelism higher than the comparison method. And then, the matricial method is better suited to an additional redundancy deletion such as quasi-redundancy than the comparison method is. In all cases, these two methods need to be used only when both theorems 7 and 8 fail.

1.7.2 Modified Fourier's Algorithm

In order to take into account the results of theorems 7 and 8, each inequation i is associated with three sets: the set H_i of initial inequations from which i is produced, the set E_i of its effectively eliminated variables, and the set I_i of its implicitly eliminated variables. When i is an initial inequation, $H_i = \{i\}$, and E_i and I_i are empty. The pattern algorithm is described in Figure 1.3.

In practice, it does not matter that some variables not implicitly eliminated occur in I_i, if each of these variables occur in at least one inequation of H_i. This comes from the fact that I_i is used only in thereom 7. So, these variables which are officially eliminated, are effectively eliminated, and then occur in E_i.

Note that steps 2.3.4 and 2.3.5 can be interchanged. Particularly, if the inequations are ordered, one can quickly see when an inequation is quasi-redundant and avoid the minimality detection for its historical subset [14].

If the strong redundancies are suppressed, (step 2.3.5), it is advisable

Input: S an inequation system $Ax \leq b$.

Let us assume that $x = (x', x'')$, where $x'' = (x_1, \ldots, x_k)$ is the vector of variables to be eliminated.

Output: A system $Cx' \leq d$ equivalent to $Ax \leq b$ on x'

begin

1. The set O of officially eliminated variables, is empty.
2. **For** x_j being successively the variables $x_1, \ldots, x_k$,
   - 2.1. Suppress from S all the inequations in which the coefficient of x_j is non-zero.
   - 2.2. If there is at least one pair of suppressed inequations with opposite sign coefficients of x_j, put x_j into O (*Duffin improvement*), and continue to 2.3, otherwise continue to 2.
   - 2.3. **For each** pair of suppressed inequations $a_1 x \leq b_1$ and $a_2 x \leq b_2$, of which the coefficients $a_{j,1}$ and $a_{j,2}$ of x_j are respectively positive and negative,
     - 2.3.1 Assume that the sets associated with these inequations are respectively H_1, E_1, I_1 and H_2, E_2, I_2. Compute $H = H_1 \bigcup H_2$, $E = E_1 \bigcup E_2$ and $I = I_1 \bigcup I_2$.
     - 2.3.2 If $Card(E \bigcup (I \bigcap O)) < Card(H)$ then continue to 2.3. (*Theorem 7*).
     - 2.3.3 If $Card(E) = Card(H)$ then go to 2.3.5. (*Theorem 8*).
     - 2.3.4 Analyze using comparison [3] or matricial computation the set H. If it is a minimal part then go to 2.3.5, else continue to 2.3.
     - 2.3.5 Suppress some strongly redundant inequations (*Theorem 9*). If the new inequation is strongly redundant then continue to 2.3, else continue to 2.3.6.
     - 2.3.6 Put in S the inequation
       $$a_{j,1} a_2 x - a_{j,2} a_1 x \leq a_{j,1} b_2 - a_{j,2} b_1.$$
3. Return the system S which is then of the form $Cx' \leq d$.

end

[3] When the comparison method is chosen, also suppress from the system the inequations of which the historical subsets include that of the new inequation.

Figure 1.3
Pattern Modified Fourier's Algorithm

to use the matricial computation at step 2.3.4.

1.8 An example

In the following example, each inequation is associated with a triplet $(H;\ E;\ I)$. H is the historical subset of the inequation, E the set of its implicitly eliminated variables and I the set of its implicitly eliminated variables. In the following notation, a hyphen will denote the empty-set, and a non-empty set will be denoted by the juxtaposition of its elements separated by dots. Let x_i, $i = 1, \ldots, 5$ be the variables to be eliminated in the following system:

(1)	$0 \leq -1x_3 - 1x_4 - 1x_5 + 1$	$(1;\ -;\ -)$
(2)	$0 \leq +1x_1 + 2x_4 - 1y_2 + 2$	$(2;\ -;\ -)$
(3)	$0 \leq +1x_2 + 2x_5 - 1y_3 + 2$	$(3;\ -;\ -)$
(4)	$0 \leq -2x_2 - 3x_5 + 1y_3 - 1$	$(4;\ -;\ -)$
(5)	$0 \leq +1x_2$	$(5;\ -;\ -)$
(6)	$0 \leq +1x_3$	$(6;\ -;\ -)$
(7)	$0 \leq -1x_1 - 1x_2 + 2x_3 - 1y_1 + 3$	$(7;\ -;\ -)$
(8)	$0 \leq -1x_1 - 1x_2 - 2x_3 - 2x_4 - 2x_5 + 1y_1 + 1y_2 + 1y_3 - 5$	$(8;\ -;\ -)$
(9)	$0 \leq -1x_2 - 1x_5 + 2y_2$	$(9;\ -;\ -)$

The elimination of x_1 gives $O_1 = \{x_1\}$, suppresses the inequations (2), (7) and (8), and proposes two pairs to create new inequations. These two new inequations are all accepted as a result of theorem 8. The cost is then minimal. The new system is:

$0 \leq -1x_3 - 1x_4 - 1x_5 + 1$	$(1;\ -;\ -)$
$0 \leq +1x_2 + 2x_5 - 1y_3 + 2$	$(3;\ -;\ -)$
$0 \leq -2x_2 - 3x_5 + 1y_3 - 1$	$(4;\ -;\ -)$
$0 \leq +1x_2$	$(5;\ -;\ -)$
$0 \leq +1x_3$	$(6;\ -;\ -)$
$0 \leq -1x_2 - 1x_5 + 2y_2$	$(9;\ -;\ -)$
$0 \leq -1x_2 + 2x_3 + 2x_4 - 1y_1 - 1y_2 + 5$	$(2.7;\ x_1;\ -)$
$0 \leq -1x_2 - 2x_3 - 2x_5 + 1y_1 + 1y_3 - 3$	$(2.8;\ x_1;\ x_4)$

The elimination of x_2 gives $O_2 = \{x_1, x_2\}$, suppresses six inequations, and proposes eight pairs to create new inequations. These eight new

inequations are all accepted as a result of theorem 8. The new system is:

$0 \leq -1x_3 - 1x_4 - 1x_5 + 1$	$(1;\ -;\ -)$
$0 \leq +1x_3$	$(6;\ -;\ -)$
$0 \leq +1x_5 - 1y_3 + 3$	$(3.4;\ x_2;\ -)$
$0 \leq +1x_5 + 2y_2 - 1y_3 + 2$	$(3.9;\ x_2;\ -)$
$0 \leq +2x_3 + 2x_4 + 2x_5 - 1y_1 - 1y_2 - 1y_3 + 7$	$(2.3.7;\ x_1.x_2;\ -)$
$0 \leq -2x_3 + 1y_1 - 1$	$(2.3.8;\ x_1.x_2;\ x_4.x_5)$
$0 \leq -3x_5 + 1y_3 - 1$	$(4.5;\ x_2;\ -)$
$0 \leq -1x_5 + 2y_2$	$(5.9;\ x_2;\ -)$
$0 \leq +2x_3 + 2x_4 - 1y_1 - 1y_2 + 5$	$(2.5.7;\ x_1.x_2;\ -)$
$0 \leq -2x_3 - 2x_5 + 1y_1 + 1y_3 - 3$	$(2.5.8;\ x_1.x_2;\ x_4)$

So far, each pair produces a retained inequation. The elimination of x_3 gives $O_3 = \{x_1, x_2, x_3\}$ and proposes nine pairs for the creation of inequations. Among these nine pairs, two are rejected according to theorem 7. The other seven pairs are accepted as a result of theorem 8. The new system is then:

$0 \leq +1x_5 - 1y_3 + 3$	$(3.4;\ x_2;\ -)$
$0 \leq +1x_5 + 2y_2 - 1y_3 + 2$	$(3.9;\ x_2;\ -)$
$0 \leq -3x_5 + 1y_3 - 1$	$(4.5;\ x_2;\ -)$
$0 \leq -1x_5 + 2y_2$	$(5.9;\ x_2;\ -)$
$0 \leq -1x_4 - 1x_5 + 1$	$(1.6;\ x_3;\ -)$
$0 \leq +1y_1 - 1$	$(2.3.6.8;\ x_1.x_2.x_3;\ x_4.x_5)$
$0 \leq -2x_5 + 1y_1 + 1y_3 - 3$	$(2.5.6.8;\ x_1.x_2.x_3;\ x_4)$
$0 \leq -1y_1 - 1y_2 - 1y_3 + 9$	$(1.2.3.7;\ x_1.x_2.x_3;\ x_4.x_5)$
$0 \leq +2x_4 + 2x_5 - 1y_2 - 1y_3 + 6$	$(2.3.7.8;\ x_1.x_2.x_3;\ x_4.x_5)$
$0 \leq -2x_5 - 1y_1 - 1y_2 + 7$	$(1.2.5.7;\ x_1.x_2.x_3;\ x_4)$
$0 \leq +2x_4 - 2x_5 - 1y_2 + 1y_3 + 2$	$(2.5.7.8;\ x_1.x_2.x_3;\ x_4)$

The elimination of x_4 gives $O_4 = \{x_1, x_2, x_3, x_4\}$ and proposes two pairs for the creation of inequations. According to theorem 7 no new inequation is created. The new system is then:

$$
\begin{array}{ll}
0 \leq +1x_5 - 1y_3 + 3 & (3.4;\ x_2;\ -) \\
0 \leq +1x_5 + 2y_2 - 1y_3 + 2 & (3.9;\ x_2;\ -) \\
0 \leq -3x_5 + 1y_3 - 1 & (4.5;\ x_2;\ -) \\
0 \leq -1x_5 + 2y_2 & (5.9;\ x_2;\ -) \\
0 \leq +1y_1 - 1 & (2.3.6.8;\ x_1.x_2.x_3;\ x_4.x_5) \\
0 \leq -2x_5 + 1y_1 + 1y_3 - 3 & (2.5.6.8;\ x_1.x_2.x_3;\ x_4) \\
0 \leq -1y_1 - 1y_2 - 1y_3 + 9 & (1.2.3.7;\ x_1.x_2.x_3;\ x_4.x_5) \\
0 \leq -2x_5 - 1y_1 - 1y_2 + 7 & (1.2.5.7;\ x_1.x_2.x_3;\ x_4)
\end{array}
$$

The elimination of x_5 gives $O_5 = \{x_1, x_2, x_3, x_4, x_5\}$ and proposes eight pairs for the creation of inequations. Two are accepted according to theorem 8. Two are rejected according to theorem 7. And four are rejected using comparison or matricial computation methods. The final system is:

$$
\begin{array}{ll}
0 \leq +1y_1 - 1 & (2.3.6.8;\ x_1.x_2.x_3;\ x_4.x_5) \\
0 \leq -1y_1 - 1y_2 - 1y_3 + 9 & (1.2.3.7;\ x_1.x_2.x_3;\ x_4.x_5) \\
0 \leq -1y_3 + 4 & (3.4.5;\ x_2.x_5;\ -) \\
0 \leq +4y_2 - 1y_3 + 2 & (3.5.9.;\ x_2.x_5;\ -)
\end{array}
$$

Let us compare, on this small example, the cost of the Černikov-Fourier and Kholer-Fourier Algorithms using Theorem 5 (column entitled "Černikov/Kohler" in the table below), and the cost of these algorithms by replacing Theorem 5 by Theorem 7 and Theorem 8 (column entitled "+ Theorems 7 and 8"). The detailed account of realized operations is given in the table below. In the Černikov or Kholer methods column, the first number represents the usage number of Theorem 5, the right number is the usage number of comparison or matricial method. In the right column, the first number represents the usage number of Theorem 7, the second number represents the usage number of Theorem 8, and the third number represents the usage number of the comparison or matricial method.

step	Černikov/Kohler	+ Theorems 7 and 8
1	0 / 2	0 / 2 / 0
2	0 / 8	0 / 8 / 0
3	2 / 7	2 / 7 / 0
4	2 / 0	2 / 0 / 0
5	0 / 8	2 / 2 / 4
total	4 / 25	6 / 19 / 4

It can be seen, in this example, that the proportions of heavy detection methods used, are completely exchanged depending on the use of theorem 7 and 8: (4/29), or theorem 5: (25/29). The advantage supplied by the two acceleration theorems is greater insomuch as the number of implicitly eliminated variables is lower. The extreme case is the one where this number is zero. In this extreme case, only these two theorems are needed for all decisions. This is generally the case when the coefficients of the constraints are randomly generated.

1.9 Conclusion

After presenting the approaches of Černikov and Kohler for improving the Fourier elimination algorithm, we have shown that these two approaches produce exactly the same final inequation system. The use of theorems 7 and 8 introduced in [10] considerably decreases the use of costly minimal subset detection methods. We have seen that the matricial minimal subset detection method is better suited to an additional redundancy deletion than the comparison detection method.

In addition, we have tested the heuristic which eliminates, at each Fourier step, the variable with the least production of inequations. The results obtained are clearly not as good as with any other choice, even at random. In some cases, the computation time increases from half a minute to more than an hour, with an overwhelming production. The lower initial production can quickly become disastrous.

Finally, the generalization of the passive variable concept to variables which only produce non-retained constraints, is incorrect, as is shown in the following example. Let the initial system be:

$$\begin{array}{lll} (1) & 0 \leq +x+y+z+t & (1;\,-;\,-) \\ (2) & 0 \leq -x-y+z & (2;\,-;\,-) \\ (3) & 0 \leq +x-y-z & (3;\,-;\,-) \\ (4) & 0 \leq -x+y-z & (4;\,-;\,-) \end{array}$$

The elimination of x gives

$$\begin{array}{ll} 0 \leq +2z+t & (1.2;\,x;\,y) \\ 0 \leq -2z & (3.4;\,x;\,y) \\ 0 \leq +2y+t & (1.4;\,x;\,z) \\ 0 \leq -2y & (2.3;\,x;\,z) \end{array}$$

The elimination of y gives the system

$$\begin{array}{ll} 0 \leq +2z + t & (1.2;\ x;\ y) \\ 0 \leq -2z & (3.4;\ x;\ y) \end{array}$$

and the elimination of z gives the final system

$$0 \leq t \quad (1.2.3.4;\ x.z;\ y).$$

If we had not taken the variable y into account, the inequation $0 \leq t$ would be rejected. Then the final system would be incorrect. This also shows the importance of the implicitly eliminated variables.

For new improvements of the Fourier algorithm, it would be interesting to see under which conditions the weakly redundant inequations (the ones for which the supporting hyperplanes are adjacent to the solution set of the inequation system) can be deleted without putting the previous improvements of the Fourier elimination on the wrong track. Whatever the future improvements however, we cannot prevent the Fourier elimination from producing a great deal of constraints, with a significant increase in the intermediary steps, before this number decreases when few variables remain in the system.

Acknowledgements

This research was supported by the ACCLAIM Esprit Research Project (PE7195).

We are grateful to David Wonnacott who have read and commented on earlier version of this paper. I would also like to thank Pamela Morton for her help in the English correctness of this paper.

Bibliography

[1] S.N. Černikov. Contraction of Systems of Linear Inequalities. In *Soviet mathematics* DOKLADY 1, 1960.

[2] S.N. Černikov. The Solution of Linear Programming Problem by Elimination of Unknowns. In *Soviet mathematics* DOKLADY 2, 1961.

[3] S.N. Černikov. Contraction of Finite Systems of Linear Inequalities. In *Soviet mathematics* DOKLADY 4, 1963.

[4] A. Colmerauer. Opening the Prolog III Universe. In *BYTE*, p177–182, August 1987.

[5] A. Colmerauer. An Introduction to Prolog III. In *Communications of the ACM*, 33, vol 7, July 1990.

[6] R.J. Duffin. On Fourier's Analyse of Linear Inequality Systems. In *Mathematical Programming Study* 1 71–95. North-Holland Publishing Company. (1974)

[7] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The Constraint Logic Programming Language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, December 1988.

[8] J.B.J Fourier. reported in : Analyse des travaux de l'Académie Royale des Sciences, pendant l'année 1824, Partie mathématique. *Histoire de l'Académie Royale des Sciences de l'institut de France 7* 1827, xlvii-lv. (Partial English translation in: D.A. Kohler, Translation of a Report by Fourier on his work on Linear Inequalities, *Opsearch 10* pages 38–42, 1973).

[9] JL. Imbert. Simplification des Systèmes de Contraintes Numériques Linéaires. *Thèse de Doctorat de l'Université d'Aix-Marseille II*, faculté des Sciences de Luminy, Mai 1989.

[10] JL. Imbert. About Redundant Inequalities Generated by Fourier's Algorithm. In P. Jorrand, editor, *Proceedings of the Fourth International Conference on Artificial Intelligence*, AIMSA'90, p 117-127.Varna, Bulgaria. North-Holland. 1990

[11] JL. Imbert and P. Van Hentenryck. A Note on Redundant Linear Constraints. *Technical Report CS-92-11*, CS Department, Brown University, 1992. 13 pages.

[12] J.Jaffar and JL. Lassez. Constraint Logic Programming. *Technical Report 86/73*. Dept. of computer science. Manash University (June 1986). An abstract appears in *Proceedings of the 14th Principles of Programming Languages*. Munich. p 111–119, 1987.

[13] J.Jaffar, S. Michaylov. Methodology and Implementation of a CLP System. In *Proceedings of the Logic Programming Conference*. Melbourne, M.I.T. Press, 1987.

[14] J.Jaffar, M.J. Maher, P.J. Stuckey and R.H.C. Yap. Output in CLP($\Re$). In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 987–995, Tokyo, Japan. June 1992,

[15] P.C. Kanellakis, G.M. Kuper and P.Z. Revesz. Constraint Query Languages. in *Proceedings of the ACM Conference on Principles of Database Systems*. Nashville 1990.

[16] M.H Karwan, V. Lofti, J. Telgen and S. Zionts. Redundancy in Mathematical Programming: a State-fo-the-Art survey. In *Lecture Notes in Computer in Economics and Mathematical Systems*, Vol 206, springer Verlag 1983.

[17] D.A. Kohler. Projection of Convex Polyhedral Sets. *PhD Thesis*, University of California, Berkeley, 1967.

[18] C. Lassez and JL. Lassez. Quantifier Elimination for Conjunctions of Linear Constraints via a Convex Hull Algorithm. To Appear 1991.

[19] JL. Lassez, T. Huynh and K. McAloon. Simplification and Elimination of Redundant Arihtmetic Constraints. In *Constraint Logic Programming: Selected Research.*, edited by F. Benhamou and A. Colmerauer. MIT Press, 73–87 Sept 1993.

[20] J. Telgen. Redundancy and Linear Programming. *Mathematical Center Tracts 137*, Mathematisch Centrum, Amsterdam, 1981.

[21] A. Schrijver. Theory of Linear and Integer Programming. *Interscience Series in Discrete Mathematics and Optimization.* John Wiley & Sons, 1987.

[22] P. Van Hentenryck. Constraint Satisfaction in Logic Programming. *M.I.T. Press*, 1989.