INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Complete solving of linear Diophantine equational and inequational systems without adding variables

Farid AJILI , Evelyne CONTEJEAN

N° 0175

June 1995

PROGRAMME 2

ISSN 0249-6399

Complete solving of linear Diophantine equational and inequational systems without adding variables*

Farid AJILI** , Evelyne CONTEJEAN***

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet PROTHEO

Rapport technique n0175 — June 1995 — 23 pages

Abstract: In this report, we present an algorithm for solving *directly* linear Diophantine systems of both equations and inequations. Here directly means without adding slack variables for encoding inequalities as equalities. This algorithm is an extension of the algorithm due to Contejean and Devie [10] for solving linear Diophantine systems of equations, which is itself a generalization of the algorithm of Fortenbacher [7] for solving a single linear Diophantine equation. All the nice properties of the algorithm of Contejean and Devie are still satisfied by the new algorithm: it is complete, *i.e.* provides a (finite) description of the set of solutions, it can be implemented with a *bounded* stack, and it admits an incremental version. All of these characteristics enable its easy integration in the CLP paradigm

Key-words: constraints, unification, solver, constraint programming, linear programming.

(Résumé : tsvp)

*This work was partly supported by the European Contract SOL HCM No CHRX CT92 0053
**E-mail: Farid.Ajili@loria.fr
***E-mail: Evelyne.Contejean@lri.lri.fr

Unité de recherche INRIA Lorraine
Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

Résolution complète de systèmes d'équations et d'inéquations Diophantiennes linéaires sans ajout de variables

Résumé : Dans ce rapport, nous présentons un algorithme de résolution *directe* de systèmes Diophantiens linéaires d'équations et d'inéquations. Ici le mot directe veut dire que nous ne transformons pas les inéquations en équations par l'ajout de variables auxiliaires. Cet algorithme est une extension de celui d'E. Contejean et H. Devie [10] pour la résolution de systèmes d'équations Diophantiennes linéaires qui est lui même une généralisation de l'algorithme de Fortenbacher [7] pour la résolution d'une seule équation Diophatienne linéaire. Toutes les bonnes propriétés de l'algorithme d'E. Contejean et H. Devie sont encore satisfaites par le nouvel algorithme: il est complet, c'est-à-dire il calcule une description finie de l'ensemble de toutes les solutions, on peut l'implanter par le biais d'une pile de taille bornée, et il admet une version incrémentale. Toutes ces caractéristiques permettent son intégration dans le cadre de la programmation logique avec contraintes.

Mots-clé : Contraintes, unification, résolveur, programmation avec contraintes, programmation linéaire.

1 Introduction

Research on algorithms for solving linear inequational and equational constraints, or systems of them, has been widely investigated starting from the ancient Greeks. Such constraints arise in various areas of computer science and efficient algorithms are well-known for solving systems of linear constraints over reals, rational numbers [20, 19] and integers [6, 27]. Unfortunately, restricting the domain to the natural numbers makes the problem much more difficult and the algorithms in the previous class are no longer suitable.

In the recent past, several works, related to the automatic deduction framework, have shown the key role of solving *systems of linear Diophantine* [1] *equations* for many important unification problems: unification modulo associativity [23], modulo associativity and commutativity [28, 16, 21, 5], modulo distributivity [9]. Hence solving such systems has been widely investigated, and a large number of algorithms (see [3] for a survey) have been proposed by numerous authors: G. Huet [17], J. L. Lambert [22], M. Clausen & A. Fortenbacher [7], E. Contejean & H. Devie [10], J. F. Romeuf [25], A. P. Tomás & M. Filgueiras [14], L. Pottier [24] and E. Domenjoud [11, 12]. All these algorithms compute a basis, *i.e* a finite subset of solutions which provides a finite and complete representation of the set all solutions: any non-negative solution is an $\mathbb{N}$-linear combination of the solutions of the basis. It should be noticed that in the case of systems of equations, the basis of solutions is the set of *minimal* solutions. However, in [13] a kind of parametric representation is computed.

As for inequations, they are ubiquitous in several domains such as constraint logic programming (CLP), integer linear programming, and operational research. In the above literature, the algorithms for solving linear inequations over natural numbers are not complete but over finite domains [18] and they usually proceed by turning inequations into equations by introducing new variables generally called *slack* [26]. Such methods yield voluminous problems, which is an handicap since the solving complexity is an exponential in the number of variables.

It is therefore quite natural to investigate an appropriate solver for systems of linear constraints $AX = 0 \wedge BX \leq 0$ over natural numbers, which outputs a complete and a finite representation of the set of all non-negative solutions and avoids adding new variables. One year ago, Ajili has proposed such a solver for the case of a single inequation [4].

The set of all solutions of the system of linear Diophantine constraints $AX = 0 \wedge BX \leq 0$ where $X \in \mathbb{N}^q$ is an additive submonoïd of $\mathbb{N}^q$ finitely generated by the subset of *non-decomposable* solutions. An algorithm which computes such a subset is an adequate and complete one. Since deciding the decomposability of a solution is not as easy as deciding the minimality (*cf.* the case of equations), we avoid the decomposability tests by using the following remark: a solution X_0 of $AX = 0 \wedge BX \leq 0$ is *non-decomposable* if and only if $(X_0, -BX_0)$ is a *minimal* solution of the system of equations $AX = 0 \wedge BX + Z = 0$. Considering the tuple of new additional variables Z as a function of X (*i.e.* $Z = -BX$) and not as a tuple of plain variables enables us to avoid having to introduce and manipulate

[1] *i.e.* over natural numbers

them explicitly. The same idea can be applied to improve the solving of equational system of the form $BX + Z = 0$, by removing the variables Z and solving $BX \leq 0$.

Inspired by the above ideas, in this report we give a complete solver for homogeneous linear Diophantine systems of both equations and inequations. This solver can be extended for solving heterogeneous systems in the same way as the complete solvers for systems of equations [7, 9]. Thanks to its flexibility (solving inequations together with equations, extension to the heterogeneous case, incrementality), this new solver has a wide range of potential applications: it can be integrated in the CLP paradigm thanks to its ability to test the satisfiability and to check constraint entailment. This work was firstly presented in [2] and will appear in [1].

2 Basic Notions

2.1 Notations

As usually $\mathbb{N}$ denotes the set of non-negative numbers and e_j denotes the j^{th} canonical tuple of $\mathbb{N}^q$, that is

$$e_j = (\underbrace{0,\ldots,0}_{(j-1)times},1,\underbrace{0,\ldots,0}_{(q-j)times}).$$

$\cdot$ denotes the scalar product of two tuples of $\mathbb{N}^q$, but it will sometimes be omitted for short.

Definition 1 (Length and Euclidean Norm) *Let $X = (x_1, \ldots, x_q)$ be a tuple in $\mathbb{N}^q$. Its length $\sum_{i=1}^{q} x_i$ is denoted by $|X|$ and its Euclidean norm $\sqrt{\sum_{i=1}^{q} x_i^2}$ is denoted by $\|X\|$.*

Definition 2 (Orderings on $\mathbb{N}^q$) *$\leq_q$ is the component-wise extension to $\mathbb{N}^q$ of the usual ordering $\leq$ defined on $\mathbb{N}$. $<_q$ is the strict ordering associated with $\leq_q$.*

Notice that $\leq_q$ is a partial ordering on $\mathbb{N}^q$. In the following, we shall freely use $\leq$ and $<$ instead of $\leq_q$ and $<_q$ if there is no ambiguity.

Definition 3 (Linear Diophantine Systems) *A linear Diophantine system of m constraints in q unknowns can be written thanks to an $m \times q$ matrix $C = (c_{ij})_{1\leq i\leq m, 1\leq j\leq q}$ and an m-tuple $d = (d_i)_{1\leq i\leq m}$ ($c_{ij}, d_i \in \mathbb{Z}$) as follows:*

$$CX \prec d,$$

where $\prec$ belongs to $\{=, \leq\}^q$, and X is the q-tuple of unknowns.

By reindexing the lines of C, the system $CX \prec d$ can be decomposed into two parts, the equational one and the inequational one as follows $AX = a \wedge BX \leq b$. m_A and m_B are respectively the number of lines of A and B.

A non-negative solution of $CX \prec d$ is said to be non-decomposable *if it is non-null and it cannot be written as the sum of two non-null solutions.*

A linear Diophantine system $CX \prec d$ is homogeneous *when d is null.*

$\mathcal{S}ol(CX \prec 0)$ denotes the set of all non-negative solutions of $CX \prec 0$, and $\mathcal{B}as(CX \prec 0)$ denotes the set of all non-negative solutions of $CX \prec 0$ which are non-decomposable.

C_i and C^j denote respectively the i^{th} row and the j^{th} column of C.

In the following, we shall focus on homogeneous linear Diophantine systems.

2.2 Finite representation of the solutions

In general, the set $\mathcal{S}\mathrm{ol}(CX \prec 0)$ is infinite, however one can represent it in a finite way. Indeed in the homogeneous case, $\mathcal{S}\mathrm{ol}(CX \prec 0)$ is closed under addition and contains 0, hence it is an additive sub-monoïd of $\mathbb{N}^q$. By the Hilbert basis theorem [8], $\mathcal{S}\mathrm{ol}(CX \prec 0)$ is generated by a finite basis which is exactly $\mathcal{B}\mathrm{as}(CX \prec 0)$, the set of non-decomposable solutions of $CX \prec 0$. This basis provides a finite and a complete representation of $\mathcal{S}\mathrm{ol}(CX \prec 0)$, in the sense that $\mathcal{S}\mathrm{ol}(CX \prec 0)$ is the set of all $\mathbb{N}$-linear combinations of elements in $\mathcal{B}\mathrm{as}(CX \prec 0)$.

When a linear Diophantine system contains only equations, the non-decomposability of a solution coincides with its minimality *w.r.t.* $\leq_q$, but this is no longer true when the system contains also some inequations. However, one can still check the non-decomposability of a solution thanks to the following remark: the solutions of

$$AX = 0 \wedge BX \leq 0$$

are the projections over the first q components of the solutions of the system of *equations*

$$AX = 0 \wedge BX + Z = 0,$$

where Z is a m_B-tuple whose i^{th} component is the slack variable z_i. This projection is actually a one-to-one mapping from $\mathcal{S}\mathrm{ol}(AX = 0 \wedge BX + Z = 0)$ onto $\mathcal{S}\mathrm{ol}(AX = 0 \wedge BX \leq 0)$, and its inverse maps a solution $s \in \mathbb{N}^q$ to $(s, -Bs) \in \mathbb{N}^{q+m_B}$. A solution s of $AX = 0 \wedge BX \leq 0$ is non-decomposable if and only if its associated solution of $AX = 0 \wedge BX + Z = 0$ is non-decomposable, hence if and only if $(s, -Bs)$ is minimal.

3 Solving homogeneous linear Diophantine systems of equations

In '89 Contejean and Devie [10] proposed an algorithm for solving a system of several linear Diophantine equations $AX = 0$ as a whole, by computing its set of minimal solutions. It is an extension of the algorithm of Fortenbacher [7] which solves a single linear Diophantine equation. The basis ideas of both algorithms are the following:

- Search $\mathbb{N}^q \setminus \{0\}$ for the minimal solutions starting from the canonical tuples e_js.
- Suppose that the current tuple is not yet a solution. It can be non deterministically increased component by component until it becomes a solution or greater than a solution.

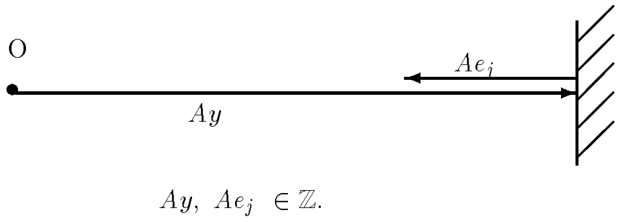


Figure 1: Geometric interpretation of Fortenbacher's restriction.

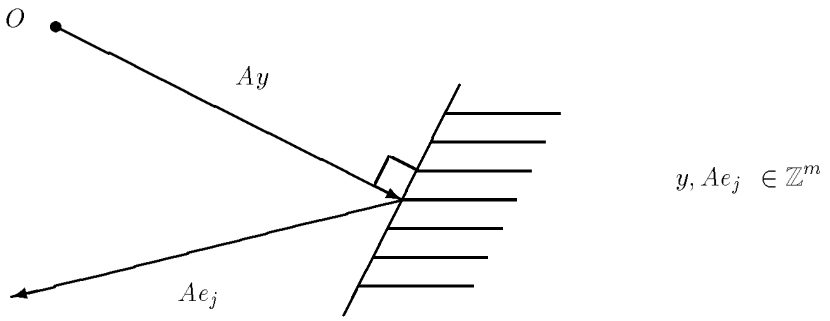


Figure 2: Geometric interpretation of Contejean and Devie's generalized restriction.

- In order to insure the termination of the search, one adds a pruning criterion which does not remove any minimal solution.

The new algorithms presented below (Subsections 4.1, 4.2) are also based on the same principles.

The pruning criterion of Fortenbacher allows a non-solution y to be incremented on its j^{th} component only if the two integers[2] Ay and Ae_j do not have the same sign.

Contejean and Devie have generalized this criterion for an arbitrary number of equations thanks to a geometrical interpretation: Ay should not become too large, hence adding Ae_j to Ay should yield a new $A(y+e_j)$ "returning to the origin", that is lying in the half-space delimited by the hyper-plan orthogonal to Ay and containing the origin. This condition can be written $Ay \cdot Ae_j < 0$.

The pruning criterion ensures the termination when $\mathbb{N}^q$ is searched breadth-first, and yields the following algorithm working on two lists, the first one being the solutions already found and the second one being the nodes already built, but not yet developed:

[2]Remember that there is only one equation.

Initialization	$[]; [e_1; \ldots; e_q]$			
Solution	$\mathcal{M}$; $[y \vert \mathcal{P}]$	$\longrightarrow$	$[y \vert \mathcal{M}]$; $\mathcal{P}$	if $Ay = 0$
Leaf	$\mathcal{M}$; $[y \vert \mathcal{P}]$	$\longrightarrow$	$\mathcal{M}$; $\mathcal{P}$	if $\exists s \in \mathcal{M}\ s \leq_q y$
Develop	$\mathcal{M}$; $[y \vert \mathcal{P}]$	$\longrightarrow$	$\mathcal{M}$; $\mathcal{P}@[y + e_{j_1}, \ldots, y + e_{j_l}]$	
	if $Ay \neq 0$, $\forall s \in \mathcal{M}\ s \not\leq_q y$			
	and $\{e_{j_1}, \ldots, e_{j_l}\} = \{e_j \mid Ay \cdot Ae_j < 0\}$			

The breadth-first version of the algorithm of Contejean and Devie
for solving $AX = 0$.

Theorem 1 (Contejean and Devie) *Given a system of m_A homogeneous linear Diophantine equations $AX = 0$, the above procedure reaches in a finite time a normal form $(\mathcal{M}; [])$, and $\mathcal{M}$ is exactly the set of minimal solutions of $AX = 0$.*

The termination proof is quite delicate and based on compactness properties and topology arguments. The soundness and completeness proofs are adapted from Fortenbacher's ones.

The breadth-first version of the algorithm is quite easy to explain and to understand, but one cannot guarantee that the size of the queue (that is the set of tuples waiting for development) is bounded. This problem is overcome by a depth-first version of the algorithm which can be implemented with a *bounded* stack.

One develops a forest (each tuple, but the roots, has exactly one father) and avoids generating some redundant nodes with a mechanism freezing some of the components of the youngest sons of a node. Assume that for each node y occurring in the graph built by the algorithm, there is a total (arbitrary) ordering $\prec_y$ on its sons. For instance, one can chose an ordering independent of y such as $y + e_{j_1} \prec_y y + e_{j_2}$ if and only if $j_1 < j_2$. If y has two distinct sons $y + e_{j_1}$, and $y + e_{j_2}$ such that $y + e_{j_1} \prec_y y + e_{j_2}$, then the j_2-th component is frozen in the sub-graph rooted at $y + e_{j_1}$: it cannot be increased any more, even if the geometrical condition expressed by the scalar product is satisfied. With such a restriction, if a node u is greater than a solution, this solution occurs in the forest at the left hand side of u. Searching the forest depth-first (from left to right) provides a complete and terminating algorithm which builds a sub-forest of the original graph. Now the size of the queue is bounded by the number of variables since a tuple at position l in the queue has exactly $q - l$ frozen components.

Here is a formal description of the depth-first version of the algorithm working on two lists, the first one being the solutions already found and the second one being the nodes already built, but not yet developed, equipped with their list of frozen components (written as an exponent):

Initialization	$[]; [e_{j_1}^{\{j_2,\dots,j_q\}}; \dots; e_{j_{q-1}}^{\{j_q\}}; e_{j_q}^{\{\}}]$		if $e_{j_1} \prec_0 \dots \prec_0 e_{j_{q-1}} \prec_0 e_{j_q}$
Solution	$\mathcal{M}$; $[y^{\mathcal{F}} \vert \mathcal{P}]$	$\longrightarrow [y \vert \mathcal{M}] ; \mathcal{P}$	if $Ay = 0$
Leaf	$\mathcal{M}$; $[y^{\mathcal{F}} \vert \mathcal{P}]$	$\longrightarrow \mathcal{M} ; \mathcal{P}$	if $\exists s \in \mathcal{M}\ s \leq_q y$
Develop	$\mathcal{M}$; $[y^{\mathcal{F}} \vert \mathcal{P}]$	$\longrightarrow \mathcal{M} ; [y + e_{j_1}^{\mathcal{F} \cup \{j_2,\dots,j_l\}}, \dots, y + e_{j_l}^{\mathcal{F}}] @ \mathcal{P}$	

if $Ay \neq 0$, $\forall s \in \mathcal{M}\ s \not\leq_q y$,
$\{j_1, \dots, j_l\} = \{j \mid Ay \cdot Ae_j < 0 \text{ and } j \notin \mathcal{F}\}$
and $y + e_{j_1} \prec_y \dots \prec_y y + e_{j_l}$

The depth-first version of the algorithm of Contejean and Devie

4 Solving linear Diophantine systems of constraints

It is well-known that solving the system of both equations and inequations

$$AX = 0 \wedge BX \leq 0$$

is equivalent to solve the system of *equations*

$$AX = 0 \wedge BX + Z = 0,$$

and then forget the variables Z by a projection. The key idea of the algorithms described below is that this will be done in a single step, without introducing explicitly the additional variables Z. The standard algorithm of Contejean and Devie will be applied, with the main difference that a q-tuple y does not only represent itself but also a set of $(q + m_B)$-tuples of the form

$$(y, u_1, \dots, u_{m_B}) \text{ such that } \forall 1 \leq i \leq m_B \quad u_i \in \{0, \dots, max(0, -B_i y)\}.$$

Hence, the algorithm for solving systems of both equations and inequations can be roughly described as follows:

- Search $\mathbb{N}^q \setminus \{0\}$ for the non-decomposable solutions starting from the canonical tuples e_js.
- Suppose that the current tuple can still provide some non-decomposable solutions. It can be non deterministically increased component by component until it cannot provide any non-decomposable solution.

- In order to speed up the search, the new pruning criterion allows y to be incremented on its j^{th} component ($1 \leq j \leq q$) only if there is a $(q + m_B)$-tuple represented by y which can be incremented on its j^{th} component according to the former criterion.

Concerning the second point, the fundamental difference between solving a system of equations and a system of both equations and inequations is that in the first case, a node y greater than a solution cannot have some non-decomposable (*i.e. minimal*) solutions as descendants, whereas it can in the second case. Of course, this is because non-decomposability is not equivalent to minimality in this latter case as can be seen on the following example.

Example 2 *Consider the inequation $x - y \leq 0$. Its set of solutions is equal to $\{(n, n + n') \mid n, n' \in \mathbb{N}\}$. This set is completely described as the $\mathbb{N}$-linear combinations of the non-decomposable solutions $(1, 1)$ and $(0, 1)$. However these solutions are comparable with $<_2$: $(0, 1) <_2 (1, 1)$. If we take into account the hidden component corresponding to the additional variable z usually introduced for turning the inequation $x - y \leq 0$ into the equation $x - y + z = 0$, $(1, 1)$ and $(0, 1)$ respectively correspond to $(1, 1, 0)$ and $(0, 1, 1)$ which are no longer comparable.*

However, we want to cut a DAG rooted at a node y as soon as possible if it does not contain any non-decomposable solution. In the case of a system of both equations and inequations $AX = 0 \wedge BX \leq 0$, a sufficient criterion is that there exists a solution s of $AX = 0 \wedge BX = 0$ such that $(s, 0) <_{q+m_B} (y, -By)$.

Every descendant $y + y'$ of y which is a solution of $AX = 0 \wedge BX \leq 0$ is decomposable into s and $y + y' - s$: by hypothesis, s is a solution, and since s is smaller than y, $y + y' - s$ is in $\mathbb{N}^q$ and moreover

$$A(y + y' - s) = A(y + y') - As = 0 - 0 = 0,$$
$$B(y + y' - s) = B(y + y') - Bs = B(y + y') - 0 \leq 0.$$

Hence $y + y' - s$ is a solution of $AX = 0 \wedge BX \leq 0$. This remark leads to split the set of $\mathcal{S}\mathrm{ol}(AX = 0 \wedge BX \leq 0)$ into two disjoint subsets, $\mathcal{S}\mathrm{ol}(AX = 0 \wedge BX = 0)$ and $\mathcal{S}\mathrm{ol}(AX = 0 \wedge BX < 0)$, and only the first one will be used for stopping the development of useless nodes.

Concerning the third point, that is the new pruning criterion, it is possible to express it in a more formal way. It is possible to increment y on its j^{th} component ($1 \leq j \leq q$) only if there is a $(q + m_B)$-tuple (y, z) represented by y which can be incremented on its j^{th} component according to the former criterion, that is

$$\exists z \in \mathbb{N}^{m_B} \quad z_i \in \{0, \ldots, \max(0, -B_i y)\} \;\wedge\; (Ay \cdot Ae_j) + ((By + z) \cdot Be_j) < 0.$$

This is equivalent to the fact that the minimal value of $(Ay \cdot Ae_j) + ((By + z) \cdot Be_j)$ w.r.t z on the domain $\mathcal{D} = \{0, \ldots, \max(0, -B_1 y)\} \times \ldots \times \{0, \ldots, \max(0, -B_{m_B} y)\}$ is negative.

This value can be computed as follows:

$$\begin{array}{c} \min_{z \in \mathcal{D}}(Ay \cdot Ae_j) + ((By+z) \cdot Be_j) \\ = \\ \min_{z \in \mathcal{D}}(\sum_{i=1}^{m_A}(A_i y A_i e_j) + (\sum_{i=1}^{m_B}((B_i y + z_i) B_i e_j) \\ = \\ (\sum_{i=1}^{m_A}(A_i y A_i e_j) + (\sum_{i=1}^{m_B} \min_{z_i \in \{0,\ldots,\max(0,-B_i y)\}}((B_i y + z_i) B_i e_j) \\ = \\ (\sum_{i=1}^{m_A}(A_i y A_i e_j) + (\sum_{i=1}^{m_B} \min(B_i y B_i e_j, \max(0, B_i y) B_i e_j)). \end{array}$$

Hence, the new pruning criterion $(\mathcal{C}1)$ can be expressed by: *It is possible to increment y on its j^{th} component ($1 \leq j \leq q$) only if*

$$\sum_{i=1}^{m_A}(A_i y A_i e_j) + (\sum_{i=1}^{m_B} min(B_i y B_i e_j, max(0, B_i y) B_i e_j)) < 0.$$

Unfortunately, in the general case we do not succeed in proving the termination with this criterion alone. We need to add another one, compatible with $(\mathcal{C}1)$, and which ensures the termination. However, we can prove the termination with $(\mathcal{C}1)$ in the case of a single inequation. The next subsection is devoted to this particular case, which is much simpler than the general case treated in the subsection 4.2.

4.1 Solving a single linear Diophantine inequation

In the case of a single inequation $B_1 X \leq 0$, the inequality used by the criterion $(\mathcal{C}1)$:

$$\sum_{i=1}^{m_A}(A_i y A_i e_j) + (\sum_{i=1}^{m_B} \min(B_i y B_i e_j, \max(0, B_i y) B_i e_j)) < 0,$$

can be rewritten in a simpler way: indeed $\min(B_1 y B_1 e_j, \max(0, B_1 y) B_1 e_j) < 0$ is equivalent to:

$$B_1 y B_1 e_j < 0.$$

This can be seen by an elementary case reasoning on the sign of $B_1 y$.

Hence, the algorithm can be formally described thanks to three lists[3], the first contains the solutions of $B_1 X = 0$, the second one the solutions of $B_1 X < 0$, and the last one the nodes to develop:

[3] In the case of systems of *equations*, two lists are enough: the list of the solutions and the list of the nodes to develop, but in the case of *inequations*, we have to split the list of solutions *cf.* the remark concerning the second point of the rough description of the general algorithm.

Initialization $[]; \quad []; [e_1; \ldots; e_q]$
Solution$_=$ $\mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow [y|\mathcal{M}_=]\ ; \mathcal{M}_< \quad ; \mathcal{P}$ if $B_1 y = 0$
Leaf $\mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \mathcal{M}_= \ ; \mathcal{M}_< \quad ; \mathcal{P}$ if $\exists s \in \mathcal{M}_=\ s \leq_q y$
Solution$_<$ $\mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \mathcal{M}_= \quad ; [y|\mathcal{M}_<]\ ; \mathcal{P}@[y+e_{j_1}, \ldots, y+e_{j_l}]$
if $\forall s \in \mathcal{M}_=\ s \not\leq_q y$,
$B_1 y < 0$, $\forall s \in \mathcal{M}_<\ (s, -B_1 s) \not\leq_{q+1} (y, -B_1 y)$
and $\{e_{j_1}, \ldots, e_{j_l}\} = \{e_j \mid B_1 y B_1 e_j < 0\}$
Develop $\mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \mathcal{M}_= \quad ; \mathcal{M}_< \quad ; \mathcal{P}@[y+e_{j_1}, \ldots, y+e_{j_l}]$
if $\forall s \in \mathcal{M}_=\ s \not\leq_q y$,
$(B_1 y \not\leq 0$ or $\exists s \in \mathcal{M}_<\ (s, -Bs) \leq_{q+1} (y, -B_1 y))$
and $\{e_{j_1}, \ldots, e_{j_l}\} = \{e_j \mid B_1 y B_1 e_j < 0\}$

The breadth-first version of the algorithm for solving a single inequation $B_1 X \leq 0$.

Theorem 3 (Soundness and completeness) *Given an inequation $B_1 X \leq 0$ the above procedure reaches in a finite time a normal form $(\mathcal{M}_=; \mathcal{M}_<; [])$, and $\mathcal{M}_= \cup \mathcal{M}_<$ is exactly the set of non-decomposable solutions of $B_1 X \leq 0$.*

Proof. We denote by $(\mathcal{C}0)$ the criterion expressed by: *It is possible to increment y on its j^{th} component $(1 \leq j \leq q)$ only if*

$$B_1 y B_1 e_j < 0$$

1. *Completeness: Let s be a non-decomposable solution of $B_1 X \leq 0$. We show that it is possible to build a sequence of tuples*

$$v_1 = e_{j_0} < v_2 = v_1 + e_{j_1} < v_3 \ldots < v_k < v_{k+1} = v_k + e_{j_k} < \ldots v_{|s|-1} < v_{|s|} = s$$

such that at each step the pruning criterion $\mathcal{C}0$ allows to add e_{j_k} to v_k and get v_{k+1}.

- $k = 1$*: we can choose as an e_{j_0} any e_j, $j \in [1..q]$, such that $e_j \leq s$.*
- $k \to k+1$*: Suppose that we have already built a sequence $v_1, \ldots, v_k$, and that $v_k < s$. We show by contradiction that there exists an e_{j_k} such that $v_k + e_{j_k} \leq s$ and $B_1 v_k B_1 e_j < 0$: let us assume that there is not such an e_{j_k}. This means that*

$$\forall e_j \quad v_k + e_j \leq s \quad \Rightarrow \quad B_1 v_k B_1 e_j \geq 0$$

$s - v_k$ can be written as $\sum_{\{j | v_k + e_j \leq s\}} \lambda_j e_j$, $\lambda_j \in \mathbb{N} \setminus \{0\}$. Hence

$$\begin{array}{rcl} B_1 v_k B_1 (s - v_k) & = & \sum_{\{j|v_k+e_j \leq s\}} \underbrace{\lambda_j}_{\geq 0} \underbrace{B_1 v_k B_1 e_j}_{\geq 0} \\ B_1 v_k B_1 (s - v_k) & \geq & 0 \end{array}$$

$B_1 v_k$ is actually an integer, hence we can perform a short case analysis on its sign. Here lies the main difference between that case of a single inequation and the general case.

- *$B_1 v_k < 0$: This fact, together with $B_1 v_k B_1(s - v_k) \geq 0$, imply that $B_1(s - v_k) \leq 0$. Hence*
$$(v_k, -B_1 v_k) < (s, -B_1 s)$$
which contradicts the hypothesis that s is a non-decomposable solution.
- *$B_1 v_k = 0$: Again, we have $(v_k, -B_1 v_k) = (v_k, 0) < (s, -B_1 s)$, which contradicts the hypothesis that s is a non-decomposable solution.*
- *$B_1 v_k > 0$: Together with $B_1 v_k B_1(s - v_k) \geq 0$, this implies that $B_1(s - v_k) \geq 0$, hence $B_1 s \geq B_1 v_k > 0$, which contradicts the hypothesis that s is a solution.*

Hence, v_{k+1} can be constructed as $v_{k+1} = v_k + e_{j_k}$.

2. *Soundness: By definition, any tuple in $\mathcal{M}_= \cup \mathcal{M}_<$ is a solution of $B_1 X \leq 0$. We have to show that any tuple in $\mathcal{M}_= \cup \mathcal{M}_<$ is a* non-decomposable *solution of $B_1 X \leq 0$. Let s be such a tuple. Suppose that s is decomposable.*

   - *If $B_1 s < 0$, there exists a non-decomposable solution s', such that $(s', -B_1 s') < (s, -B_1 s)$ and $B_1 s' < 0$. Since the procedure is complete, and $s' < s$, when the rule* **Solution$_<$** *adds s to $\mathcal{M}_<$, $\mathcal{M}_<$already contains s'. This is in contradiction with the condition $\forall r \in \mathcal{M}_< \quad (r, -B_1 r) \not\leq (s, -B_1 s)$.*
   - *If $B_1 s = 0$, let us consider a sequence of tuples built by the procedure*
$$v_1 = e_{j_0} < v_2 = v_1 + e_{j_1} < v_3 \ldots < v_k < v_{k+1} = v_k + e_{j_k} < \ldots v_{|s|-1} < v_{|s|} = s$$
such that at each step the pruning criterion $\mathcal{C}0$ allows to add e_{j_k} to v_k and get v_{k+1}. Since s is decomposable, $v_{|s|-1}$ is greater or equal to a non-decomposable solution s', and since $B_1 s = 0$, $B_1 s' = 0$. Before the rule **Solution$_=$** *adds s to $\mathcal{M}_=$, there is a step where $v_{|s|-1}$ is developed by the rule* **Develop**. *Since the procedure is complete, and $s' \leq v_{|s|-1}$, at this step, $\mathcal{M}_=$already contains s'. This contradicts the condition $\forall r \in \mathcal{M}_= \quad r \not\leq v_{|s|-1}$.*

   To sum up, s cannot be decomposable.

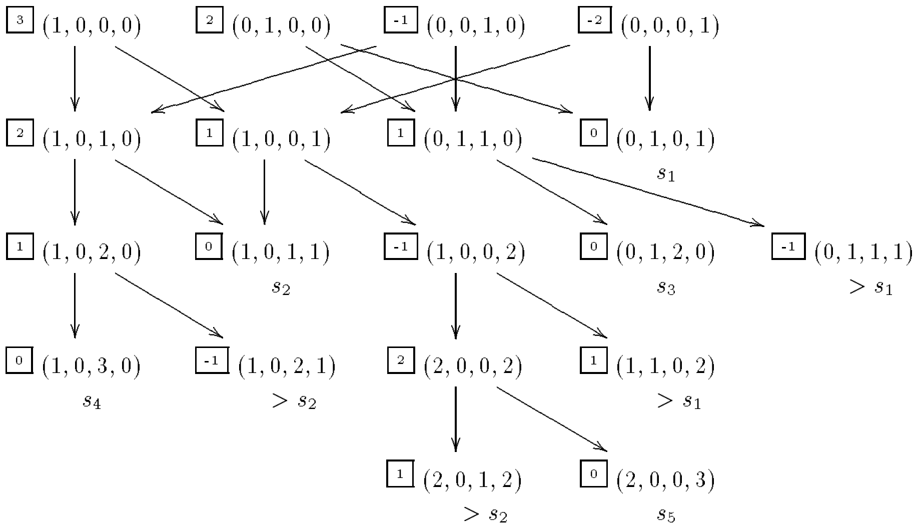
□

It should be noticed that running this algorithm for solving $B_1 X \leq 0$ yields exactly the same DAG as the algorithm of Fortenbacher for solving $B_1 X = 0$, the only difference being the way of checking the solutions: in both cases, the solutions of $B_1 X = 0$ are retained, and in the case of an inequation, the solutions of $B_1 X < 0$ are also retained but stored in a second set *not used for cutting the search-space by the rule* **Leaf**. Hence the termination of the algorithm is a corollary of the termination of the algorithm of Fortenbacher. The fact that $\mathcal{M}_= \cup \mathcal{M}_<$ contains only some solutions of $B_1 X \leq 0$ is obvious, and the proof of completeness of the criterion ($\mathcal{C}1$) (*i.e.* every non-decomposable solution is in $\mathcal{M}_= \cup \mathcal{M}_<$) will be sketched in the section devoted to the general case.

Example 4 *Consider the inequation*

$$3x_1 + 2x_2 - x_3 - 2x_4 \leq 0.$$

Running the algorithm yields the following DAG:



In the small box at the left hand side of each node $y = (y_1, y_2, y_3, y_4)$*, there is the value* $3y_1 + 2y_2 - y_3 - 2y_4$*. The non-decomposable solutions have a double frame. The set* $\mathcal{M}_=$ *of the minimal solutions of the associated equation* $3x_1 + 2x_2 - x_3 - 2x_4 = 0$ *is equal to*

$$\mathcal{M}_= = \{(0,1,0,1); (1,0,1,1); (0,1,2,0); (1,0,3,0); (2,0,0,3)\},$$

and the set $\mathcal{M}_<$ *of the other non-decomposable solutions of the inequation* $3x_1 + 2x_2 - x_3 - 2x_4 \leq 0$ *is equal to*

$$\mathcal{M}_< = \{(0,0,1,0); (0,0,0,1); (1,0,0,2)\}.$$

4.2 General case

In the case of a system containing a linear inequation together with at least another linear constraint, the pruning criterion $(\mathcal{C}1)$ is complete. This means that

Proposition 1 (Completeness of $(\mathcal{C}1)$) *Running the procedure* $\mathcal{P}roc(\mathcal{C}1)$ *on a system* $AX = 0 \wedge BX \leq 0$ *yields in a finite number of steps a triple* $(\mathcal{M}_=, \mathcal{M}_<, \mathcal{P})$ *such that the set of non-decomposable solutions is exactly* $\mathcal{M}_= \cup \mathcal{M}_<$.

$$
\begin{array}{lllll}
\textbf{Initialization} & []; \quad []; [e_1; \ldots; e_q] & & & \\
\textbf{Solution}_= & \mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow [y|\mathcal{M}_=] & ; \mathcal{M}_< & ; \mathcal{P} & \text{if } Ay = 0 \wedge By = 0 \\
\textbf{Leaf} & \mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \quad \mathcal{M}_= & ; \mathcal{M}_< & ; \mathcal{P} & \text{if } \exists s \in \mathcal{M}_= \ s \leq_q y \\
\textbf{Solution}_< & \mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \mathcal{M}_= & ; [y|\mathcal{M}_<] & ; \mathcal{P}@[y+e_{j_1}, \ldots, y+e_{j_l}] & \\
 & \text{if } \forall s \in \mathcal{M}_= \ s \not\leq_q y & & & \\
 & Ay = 0 \wedge By < 0, \forall s \in \mathcal{M}_< \ (s, -Bs) \not\leq_{q+m_B} (y, -By) & & & \\
 & \text{and } \{e_{j_1}, \ldots, e_{j_l}\} = \{e_j \mid (\mathcal{C})\} & & & \\
\textbf{Develop} & \mathcal{M}_=; \mathcal{M}_<; [y|\mathcal{P}] \longrightarrow \mathcal{M}_= & ; \mathcal{M}_< & ; \mathcal{P}@[y+e_{j_1}, \ldots, y+e_{j_l}] & \\
 & \text{if } \forall s \in \mathcal{M}_= \ s \not\leq_q y, & & & \\
 & (\neg(Ay = 0 \wedge By < 0) \text{ or } \exists s \in \mathcal{M}_< \ (s, -Bs) \leq_{q+m_B} (y, -By)) & & & \\
 & \text{and } \{e_{j_1}, \ldots, e_{j_l}\} = \{e_j \mid (\mathcal{C})\} & & &
\end{array}
$$

The procedure $\mathcal{P}\text{roc}(\mathcal{C})$ for solving a system $AX = 0 \wedge BX \leq 0$ parameterised by a pruning criterion $(\mathcal{C})$.

Proof.

1. Completeness: Let s be a non-decomposable solution of $AX = 0 \wedge BX \leq 0$. We show that it is possible to build a sequence of tuples

$$v_1 = e_{j_0} < v_2 = v_1 + e_{j_1} < v_3 \ldots < v_k < v_{k+1} = v_k + e_{j_k} < \ldots v_{|s|-1} < v_{|s|} = s$$

such that at each step the pruning criterion $\mathcal{C}1$ allows to add e_{j_k} to v_k and get v_{k+1}.

- $k = 1$: we can choose as an e_{j_0} any e_j, $j \in [1..q]$, such that $e_j \leq s$.
- $k \to k+1$: Suppose that we have already built a sequence $v_1, \ldots, v_k$, and that $v_k < s$. We show by contradiction that there exists an e_{j_k} such that $v_k + e_{j_k} \leq s$ and

$$\sum_{i=1}^{m_A}(A_i v_k A_i e_j) + (\sum_{i=1}^{m_B} \min(B_i v_k B_i e_j, \max(0, B_i v_k) B_i e_j)) < 0$$

Let us assume that there is not such an e_{j_k}. This means that

$$\forall e_j \quad v_k + e_j \leq s \quad \Rightarrow \quad \sum_{i=1}^{m_A}(A_i v_k A_i e_j) + (\sum_{i=1}^{m_B} \min(B_i v_k B_i e_j, \max(0, B_i v_k) B_i e_j)) \geq 0 \tag{1}$$

Let us define three sets of indices:

$$
\begin{array}{rcl}
\mathcal{J} & = & \{j \in [1..q] \mid v_k + e_j \leq s\} \\
\mathcal{I}1 & = & \{i \in [1..m_B] \mid B_i v_k \geq 0\} \\
\mathcal{I}2 & = & \{i \in [1..m_B] \mid B_i v_k < 0\}
\end{array}
$$

Since $s - v_k$ can be written as $\sum_{j\in\mathcal{J}} \lambda_j e_j$, where $\lambda_j \in \mathbb{N} \setminus \{0\}$, and summing over the index $j \in \mathcal{J}$ with the coefficients λ_j the inequalities 1, we get

$$Av_k A(s - v_k) + \sum_{j\in\mathcal{J}} \sum_{i=1}^{m_B} \lambda_j \min(B_i v_k B_i e_j,\ \max(B_i v_k, 0) B_i e_j) \geq 0 \tag{2}$$

Since s is a solution of $AX = 0 \wedge BX \leq 0$, $As = 0$. Hence the inequality 2 yields

$$\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}1} \lambda_j B_i v_k B_i e_j \geq \underbrace{-\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}2} \lambda_j \min(B_i v_k B_i e_j, 0) + \|Av_k\|^2}_{\geq 0} \tag{3}$$

Since $\|Av_k\|^2 \geq 0$ and $\min(B_i v_k B_i e_j, 0) \leq 0$, from 3, it is possible to deduce:

$$\sum_{i\in\mathcal{I}1} B_i v_k B_i (s - v_k) = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}1} \lambda_j B_i v_k B_i e_j \geq 0 \tag{4}$$

$$\sum_{i\in\mathcal{I}1} B_i v_k B_i s \geq \sum_{i\in\mathcal{I}1} (B_i v_k)^2 \geq 0 \tag{5}$$

From the definition of $\mathcal{I}1$ and the hypothesis that s is a solution of $AX = 0 \wedge BX \leq 0$, it follows that $B_i v_k B_i s$ is non-positive for all i in $\mathcal{I}1$. Hence for all i in $\mathcal{I}1$, $B_i v_k = 0$. To sum up,

$$\forall i \in [1..m_B] \quad B_i v_k \leq 0 \tag{6}$$

Hence $\max(B_i v_k, 0)$ can be replaced by 0 in 2 and we obtain:

$$Av_k A(s - v_k) + \sum_{j\in\mathcal{J}} \sum_{i=1}^{m_B} \lambda_j \min(B_i v_k B_i e_j, 0) \geq 0 \tag{7}$$

Since $As = 0$ and $\min(B_i v_k B_i e_j, 0) \leq 0$, $Av_k = 0$. Together with 6, this means that v_k is a solution of $AX = 0 \wedge BX \leq 0$. Moreover, this yields

$$\forall i \in [1..m_B] \quad \forall j \in \mathcal{J} \quad \min(B_i v_k B_i e_j, 0) = 0 \tag{8}$$

$$\forall i \in [1..m_B] \quad \forall j \in \mathcal{J} \quad B_i v_k B_i e_j \geq 0 \tag{9}$$

$$\forall i \in [1..m_B] \quad B_i v_k B_i (s - v_k) \geq 0 \tag{10}$$

If i is in $\mathcal{I}1$, then $-B_i v_k = 0 \leq -B_i s$. If i is in $\mathcal{I}2$ then 10 implies that $-B_i v_k \leq -B_i s$. In any case, $-B_i v_k \leq -B_i s$ holds. As a conclusion, v_k is a solution of $AX = 0 \wedge BX \leq 0$, such that $(v_k, -Bv_k) < (s, -Bs)$. This is a contradiction with the hypothesis that s is a non-decomposable solution of $AX = 0 \wedge BX \leq 0$.

2. Soundness: the proof is similar to the case of a single inequation.

□

Unfortunately, $(\mathcal{C}1)$ does not ensure the termination, there are some systems such that at any step $\mathcal{P}$,the last component of the triple handled by the procedure, is not empty. This is the case for the

Example 5 *Let us consider the system of inequations:*

$$\begin{array}{rcrcrcl} x_1 & - & 3x_2 & + & 4x_3 & \leq & 0, \\ 2x_1 & + & x_2 & - & x_3 & \leq & 0. \end{array}$$

The system of associated equations has no solutions, hence **Solution$_=$** *and* **Leaf** *will never apply, and it can be seen by induction on n that the procedure builds an infinite sequence of tuples*

$$\begin{array}{l} (0,1,0); (0,1,1); (0,2,1); (0,2,2); (0,2,3); (0,3,3); (0,4,3); (0,4,4); (0,5,4) \\ (0,5,5); \ldots; (0,n,n); (0,n+1,n); (0,n+1,n+1); \ldots \end{array}$$

As we want to obtain a terminating algorithm for solving $AX = 0 \wedge BX \leq 0$, we add a second pruning criterion ($\mathcal{C}2$) which ensures the termination. This is done in two steps. First, we consider ($\mathcal{C}2$) alone. Then, we consider ($\mathcal{C}2$) together with ($\mathcal{C}1$), and we shall prove that they are *compatible*.

The criterion ($\mathcal{C}2$) is based on the fact that the hidden part corresponding to the additional variables is bounded for the non-decomposable solutions of $AX = 0 \wedge BX \leq 0$. There are some uniform bounds on the minimal solutions of a system of equations $CX = 0$, in particular the following one proposed by Pottier [24]:

Lemma 1 (Pottier) *Let C be an $n \times q$ matrix of rank r. Every minimal solution $m = (m_1, \ldots, m_q)$ of $CX = 0$ satisfies:*

$$max_{1 \leq j \leq q} |m_j| \leq (n - r) \left(\frac{\sum_{i,j} |c_{ij}|}{r} \right)^r .$$

Corollary 1 *Let $AX = 0 \wedge BX \leq 0$ be a system of linear Diophantine constraints, and r be the rank of the matrix $\begin{pmatrix} A & 0 \\ B & I \end{pmatrix}$. Let s be a non-decomposable solution of $AX = 0 \wedge BX \leq 0$. Then the following inequalities hold:*

$$\begin{array}{rcl} max_{1 \leq i \leq m_B} |B_i s| & \leq & \mathcal{B}_2 \equiv (q + m_B - r) \left(\frac{\sum_{i,j} |a_{ij}| + \sum_{i,j} |b_{ij}| + m_B}{r} \right)^r , \\ \|Bs\|^2 & \leq & \mathcal{B}'_2 \equiv m_B {\mathcal{B}_2}^2. \end{array}$$

The criterion ($\mathcal{C}2$) is expressed by: *It is possible to increment y on its j^{th} component ($1 \leq j \leq q$) only if*

$$Ay \cdot Ae_j + By \cdot Be_j \leq \mathcal{B}'_2.$$

Proposition 2 (Completeness of ($\mathcal{C}$2)) *Running the procedure $\mathcal{P}roc(\mathcal{C}2)$ on a system $AX = 0 \wedge BX \leq 0$ yields in a finite number of steps a triple $(\mathcal{M}_=, \mathcal{M}_<, \mathcal{P})$ such that the set of non-decomposable solutions is exactly $\mathcal{M}_= \cup \mathcal{M}_<$.*

Proof. The completeness proof of $(\mathcal{C}2)$ is similar to the one of $(\mathcal{C}1)$: let us suppose that s is a non-decomposable solution of $AX = 0 \wedge BX \leq 0$, and that we have a sequence of tuples

$$e_{j_0} = v_1 < v_2 = v_1 + e_{j_1} < v_3 \ldots < v_k \leq s,$$

such that at each step the pruning criterion $(\mathcal{C}2)$ is satisfied. If v_k is not yet equal to s, we shall build v_{k+1} from v_k by adding an e_{j_k} such that $e_{j_k} \leq s - v_k$. s is a solution, hence

$$\begin{aligned}
&Av_k \cdot A(s - v_k) = -\|Av_k\|^2 \text{ since } As = 0,\\
&Av_k \cdot A(s - v_k) \leq 0.
\end{aligned}$$

$$\begin{aligned}
&B(s - v_k) + Bv_k + (-Bs) = 0,\\
&\|B(s - v_k) + Bv_k + (-Bs)\|^2 = 0,\\
&\underbrace{\|B(s-v_k)\|^2 + \|Bv_k\|^2 + \|-Bs\|^2}_{\geq 0} + 2(B(s-v_k)\cdot Bv_k + B(s-v_k)\cdot(-Bs) + Bv_k\cdot(-Bs) = 0,\\
&B(s - v_k)\cdot Bv_k + B(s - v_k)\cdot(-Bs) + Bv_k\cdot(-Bs)) \leq 0,\\
&B(s - v_k)\cdot Bv_k + Bs\cdot(-Bs) \leq 0,\\
&B(s - v_k)\cdot Bv_k \leq \|Bs\|^2,\\
&B(s - v_k)\cdot Bv_k \leq \mathcal{B}'_2.
\end{aligned}$$

$$Av_k \cdot A(s - v_k) + Bv_k \cdot B(s - v_k) \leq \mathcal{B}'_2.$$

There is necessarily a tuple $e_{j_k} \leq s - v_k$ such that $Av_k \cdot Ae_{j_k} + Bv_k \cdot Be_{j_k} \leq \mathcal{B}'_2$, otherwise the equality $Av_k \cdot A(s - v_k) + Bv_k \cdot B(s - v_k) \leq \mathcal{B}'_2$ cannot be satisfied. $(\mathcal{C}2)$ allows to add such an e_{j_k} to v_k in order to build v_{k+1}. □

Proposition 3 (Termination of $(\mathcal{C}2)$) *Running the procedure $\mathcal{P}roc(\mathcal{C}2)$ on a system $AX = 0 \wedge BX \leq 0$ terminates,* i.e. *yields in a finite number of steps a triple $(\mathcal{M}_=, \mathcal{M}_<, \emptyset)$.*

Proof. Let $\mathcal{S}up$ be the maximum of ${\mathcal{B}'_2}^2$, the $\|Ae_j\|^2$s and the $\|Be_j\|^2$s. Let

$$v_1, v_2 = v_1 + e_{j_1}, \ldots, v_k = v_{k-1} + e_{k-1}, v_{k+1} = v_k + e_{j_k}, \ldots$$

be a sequence of tuples built according to the criterion $(\mathcal{C}2)$. We will show by an induction on k that the following inequality holds:

$$\|Av_k\|^2 + \|Bv_k\|^2 \leq 4k\mathcal{S}up.$$

This is obvious when k is equal to 1. Let us assume that the inequality holds for k, then we can prove that the inequality holds for $k + 1$:

$$\|Av_{k+1}\|^2 + \|Bv_{k+1}\|^2 = \underbrace{\|Av_k\|^2 + \|Bv_k\|^2}_{4k\mathcal{S}up} + \underbrace{\|Ae_j\|^2}_{\leq \mathcal{S}up} + \underbrace{\|Be_j\|^2}_{\leq \mathcal{S}up} + 2\underbrace{(Av_kAe_j + Bv_kBe_j)}_{\leq {\mathcal{B}'_2}^2 \leq \mathcal{S}up} \leq 4(k+1)\mathcal{S}up$$

Hence one can deduce that $\begin{bmatrix} A \\ B \end{bmatrix} \frac{v_k}{k} \longrightarrow_{k\to\infty} 0$. Using the same arguments as in [10], it is possible to show that there exists a minimal solution s of $AX = 0 \wedge BX = 0$ and an integer k_0 such that

$$\forall k > k_0,\ s \leq v_k$$

Hence, an infinite sequence will never be produced by $\mathcal{P}\text{roc}(\mathcal{C}2)$, since it will be cut by **Leaf**. □

Proposition 4 (Completeness of $(\mathcal{C}1)\wedge(\mathcal{C}2)$) *Running the procedure $\mathcal{P}roc(\mathcal{C}1\wedge\mathcal{C}2)$ on a system $AX = 0 \wedge BX \leq 0$ yields in a finite number of steps a triple $(\mathcal{M}_=, \mathcal{M}_<, \mathcal{P})$ such that the set of non-decomposable solutions is exactly $\mathcal{M}_= \cup \mathcal{M}_<$.*

Proof. let us suppose that s is a non-decomposable solution of $AX = 0 \wedge BX \leq 0$, and that we have a sequence of tuples

$$e_{j_0} = v_1 < v_2 = v_1 + e_{j_1} < v_3 \ldots < v_k \leq s,$$

such that at each step $(\mathcal{C}1)\wedge(\mathcal{C}2)$ is satisfied. If v_k is not yet equal to s, we shall build v_{k+1} from v_k by adding an e_{j_k} such that $e_{j_k} \leq s - v_k$. $(\mathcal{C}1)$ is complete, hence there exists an $e_j \leq s - v_k$ such that

$$\sum_{i=1}^{m_A}(A_i v_k A_i e_j) + (\sum_{i=1}^{m_B} \min(B_i v_k B_i e_j, \max(0, B_i v_k) B_i e_j)) < 0.$$

If $Av_k \cdot Ae_j + Bv_k \cdot Be_j \leq \mathcal{B}'_2$, then we can chose e_j as e_{j_k}, otherwise, since

$$Av_k \cdot A(s - v_k) + Bv_k \cdot B(s - v_k) \leq \mathcal{B}'_2,$$

there is necessarily another $e_{j'} \leq s - v_k$ such that $Av_k \cdot Ae_{j'} + Bv_k \cdot Be_{j'} < 0$. Then we can chose $e_{j'}$ as e_{j_k}, since

$$\sum_{i=1}^{m_A}(A_i v_k A_i e_{j'}) + (\sum_{i=1}^{m_B} \min(B_i v_k B_i e_{j'}, \max(0, B_i v_k) B_i e_{j'})) \leq Av_k \cdot Ae_{j'} + Bv_k \cdot Be_{j'}.$$

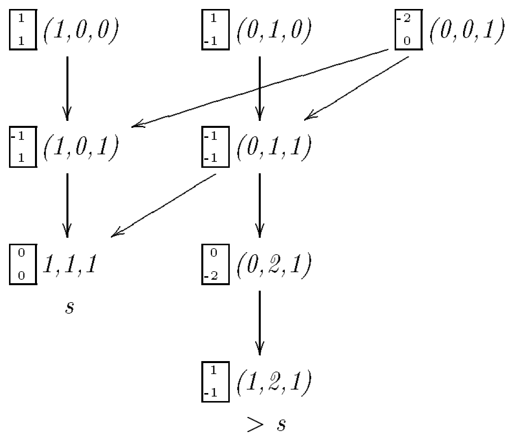
In both cases, $(\mathcal{C}1)\wedge(\mathcal{C}2)$ allows to add e_{j_k} to v_k for building $v_{j_{k+1}}$. $\square$

Hence, we have designed a conjunction of criteria $(\mathcal{C}1)\wedge(\mathcal{C}2)$ which is complete and makes the parameterised procedure terminating.

Theorem 6 *Running the procedure $\mathcal{P}roc(\mathcal{C}1\wedge\mathcal{C}2)$ on a system $AX = 0 \wedge BX \leq 0$ yields in a finite number of steps a triple $(\mathcal{M}_=, \mathcal{M}_<, \emptyset)$ such that the set of non-decomposable solutions is exactly $\mathcal{M}_= \cup \mathcal{M}_<$.*

Example 7

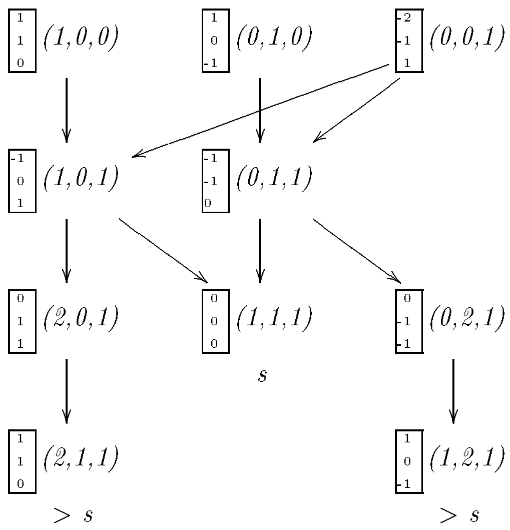
$$\begin{cases} x_1 + x_2 - 2x_3 \leq 0 \\ x_1 - x_2 \leq 0 \end{cases}$$



In this example, the criterion ($\mathcal{C}2$) (i.e. the bound) is not needed to ensure termination, it's satisfied in all the nodes. It should be noticed that solving the inequational system given above with the algorithm of E. Contejean & H. Devie (with the stack version and by adding slack variables) generates 18 nodes.

Example 8

$$\begin{cases} x + y - 2z = 0 \\ x - z = 0 \\ -y + z \leq 0 \end{cases}$$



The unique non-decomposable solution of the system is $(1,1,1)$. *Again, the criterion ($\mathcal{C}2$) does not cut any sub-DAG. The algorithm of E. Contejean & H. Devie builds a DAG of 12 nodes for the same constraint system.*

5 Some extensions

A depth-first version of the algorithm

As in the case of systems of equations, the algorithm admits a depth-first version based on a freezing mechanism together with a total ordering for the set of sons of each node. The key point here is that we only use the solutions of associated equational system for stopping the development of useless nodes, and that if a node is greater than an "equational" solution, this solution is at its left hand side in the DAG.

Solving non-homogeneous linear Diophantine constraints

The representation of the solutions of $AX = a \wedge BX \leq b$ is a bit more complicated than in the homogeneous case since its set of solutions is no longer a monoïd. Let $\mathcal{S}_0$ be the set of the non-decomposable solutions of the homogeneous system $AX = 0 \wedge BX \leq 0$ and $\mathcal{S}_1$ be the set $\{s \mid (s,1) \in \mathcal{B}\text{as}(AX - az = 0 \wedge BX - bz \leq 0)\}$. Any solution of $AX = a \wedge BX \leq b$ is a sum of a solution in $\mathcal{S}_1$ and an $\mathbb{N}$-linear combination of solutions in $\mathcal{S}_0$. Hence the "homogeneous" solver may be extended to the heterogeneous case according to the lines of T. Guckenbiehl & A. Herold [15] and E. Contejean [10]. The sets $\mathcal{S}_0$ and $\mathcal{S}_1$ can be obtained by solving the homogeneous system $AX - az = 0 \wedge BX - bz \leq 0$ by the latter algorithm and freeze the new variable for each node as soon as it's equal to 1. For any (s, z) in the set $\mathcal{M}_= \cup \mathcal{M}_<$ returned by the algorithm, we test its last component: if $z = 0$ (resp $z = 1$) then s belongs to $\mathcal{S}_0$ (resp $\mathcal{S}_1$).

6 Some potential applications

The solver in a CLP setting

In the constraint logic programming framework, the ability of testing the constraints' entailment and the satisfiability/unsatisfiability of a set of constraints (and eventually exhibiting a solution) is a crucial issue. Propagation based solvers, as for instance in the finite domains' case, are generally not highly efficient for testing constraint entailment and unsatisfiability because propagation only reasons with local consistency [29]. The solver presented in this paper provides some effective means to decide unsatisfiability since it is complete and terminates. Moreover, it also provides an algorithm for deciding entailment.

A constraint c is *entailed* by a system of constraints C if any solution of C also satisfies c. Entailment is used in the cc(FD) frame [30] for reducing an implication constraint $c \rightarrow C$ to C if c is entailed by the constraints' store. If $\neg c$ is entailed, then $c \rightarrow C$ is reduced to *True*. Given a constraint system $AX = 0 \wedge BX \leq 0$, the proposed solver returns the basis of solutions $\{m_1, .., m_u\}$. Any inequational constraint (or system of them) $\underline{\alpha} \cdot X \leq 0$, where $\underline{\alpha},\ X \in \mathbb{Z}^q$, is entailed by $AX = 0 \wedge BX \leq 0$ if and only if for all $i \in [1..u] : \underline{\alpha} \cdot m_i \leq 0$ holds. Indeed, any non-negative solution of $AX = 0 \wedge BX \leq 0$ is a $\mathbb{N}$-linear combination of $\{m_1, .., m_u\}$. The strength of this procedure lies in the fact that it provides entailment in the presence of infinite solution sets.

Solving some linear programs

Let's take the integer linear program:

$$\mathcal{LP} = \begin{cases} BX \leq 0 \\ X > 0 \\ Minimise(\underline{c}X) \\ \underline{c} = (c_i)_{1 \leq i \leq q} \mid \forall i\ c_i \in \mathbb{N}^* \end{cases}$$

In order to solve $\mathcal{LP}$, some linear programming methods are characterised by constructing a sequence of trial solutions that go through the interior of the solution space until reaching an optimal one. Hence, the disadvantage of such methods is the need for a feasible solution to start. If we note that the optimal solution of $\mathcal{LP}$ must be a minimal one of $BX \leq 0$, then the solver, presented here, can be applied to compute all the minimal solutions of $BX \leq 0$ and then, the one which minimises the objective function $\underline{c}X$ can be chosen.

7 Conclusion

We gave the *first* algorithm which computes the basis of a linear Diophantine system of both equations and *inequations* without explicitly adding some extra variables. This algorithm is actually an extension of the algorithm of E. Contejean & H. Devie, the main difference between them being the pruning criterion. Hence the new algorithm inherits all the nice characteristics of the former one: its depth-first version can be implemented with a *bounded* stack, it is incremental and compatible with the propagation traditionally used by FD solvers.

In the future, we are willing to investigate the potential applications of the new solver. This seems to be promising since many combinatorial and optimisation problems (scheduling, planning, data dependence analysis...) can be formulated as: $AX = 0 \wedge BX \leq 0$. An other interesting further work is how can this solver make use of local propagation techniques in order to prune as much as possible the search space (in our context the data structure DAG). In particular, how can one design a cooperation between the latter algorithm and a finite domain solver.

References

[1] Farid Ajili and Evelyne Contejean. Complete solving of linear Diophantine equations and inequations without adding variables. In U. Montanari and F. Rossi, editors, *Proc. of the first international conference on Principles and Practice of Constraint Programming*, pages 1–17, volume 976 of Lecture Notes in Computer Science. Cassis, France, September 1995.

[2] Farid Ajili and Contejean Evelyne. Complete solving of linear diophantine equations and inequations without adding slack variables. In Nieuwenhuis R., editor, *Proceedings of the ninth UNIF Workshop*, Barclona, Spain, April 1995.

[3] F. Ajili, E. Contejean, E. Domenjoud, M. Filgueiras, C. Kirchner, and A.-P. Tomás. Solving Linear Diophantine Equations: The State of the Art. in preparation, 1995.

[4] Farid Ajili. Etude de la résolution de contraintes diophantiennes linéaires sur les entiers naturels. Rapport de dea, Université Henri Poincaré - Nancy 1, September 1994.

[5] Alexandre Boudet, Evelyne Contejean, and Hervé Devie. A new AC-unification algorithm with a new algorithm for solving diophantine equations. In *Proc. 5th IEEE Symp.*

Logic in Computer Science, Philadelphia, pages 289–299. IEEE Computer Society Press, June 1990.

[6] T. J. Chou and G. E. Collins. Algorithms for the solution of systems of linear diophantine equations. *SIAM Journal on computing*, 11:687–708, 1982.

[7] M. Clausen and A. Fortenbacher. Efficient solution of linear diophantine equations. *Journal of Symbolic Computation*, 8(1&2):201–216, 1989.

[8] A. H. Clifford and G. B. Preston. *The algebraic theory of semigroups*. Number 7 in Mathematical surveys. American Mathematical Society, 1961. There's two volumes. The second was published in 1967.

[9] Evelyne Contejean. Solving $*$-problems modulo distributivity by a reduction to $AC1$-unification. *Journal of Symbolic Computation*, 16:493–521, 1993.

[10] Evelyne Contejean and Hervé Devie. An efficient algorithm for solving systems of diophantine equations. *Information and Computation*, 113(1):143–172, August 1994.

[11] Eric Domenjoud. Outils pour la déduction automatique dans les théories associatives-commutatives. Thèse de doctorat de l'université de Nancy I, 1991.

[12] Eric Domenjoud. Solving systems of linear diophantine equations: An algebraic approach. In *Proc. 16th Mathematical Foundations of Computer Science, Warsaw, LNCS 520*. Springer-Verlag, 1991.

[13] E. Domenjoud and A. P. Tomás. From Elliot-MacMahon to an algorithm for general linear constraints on naturals. In U. Montanari and F. Rossi, editors, *Proc. of the first international conference on Principles and Practice of Constraint Programming*, pages 18–35, volume 976 of Lecture Notes in Computer Science. Cassis, France, September 1995.

[14] M. Filgueiras and A. P. Tomás. Fast methods for solving linear diophantine equations. In M. Filgueiras and Damas L., editors, *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, 727, pages 297–306. Lecture Notes in Artificial Intelligence, Springer-Verlag, 1993.

[15] T. Guckenbiehl and A. Herold. Solving linear diophantine equations. Technical Report SEKI-85-IV-KL, 1985.

[16] Alexander Herold and Jorg H. Siekmann. Unification in abelian semi-groups. *Journal of Automated Reasoning*, 3(3):247–283, 1987.

[17] Gérard Huet. An algorithm to generate the basis of solutions to homogeneous linear diophantine equations. *Information Processing Letters*, 7(3), April 1978.

[18] E. Joxan, M. J. Maher, P. J. Stuckey, and R. H. C. Yap. Beyond finite domains. In A. Borning, editor, *Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming*, volume 874 of *Lecture Notes in Computer Science*, pages 86–94. Springer-Verlag, may 1994.

[19] N. Karmarkar. A new polynomial algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, New York, 1984. Revised version: Combinatorica 4 (1984),373-395.

[20] L. G. Khachiyan. Polynomial algorithms in linear programming. *Zhurnal Vychisditel'noi Matematiki i Matematicheskoi Fiziki*, pages 51–68, 1980.

[21] Claude Kirchner. From unification in combination of equational theories to a new AC unification algorithm. In H. Ait-Kaci and M. Nivat, editors, *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, pages 171–210. Academic Press, 1987.

[22] J. L. Lambert. Une borne pour les générateurs des solutions entières positives d'une équation diophantienne linéaire. *Comptes Rendus de l'Académie des Sciences de Paris*, 305:39,40, 1987. Série I.

[23] G. S. Makanin. Algorithmic decidability of the rank of constant free equations in a free semigroup. *Dokl. Akad. Nauk. SSSR 243*, 243, 1978.

[24] L. Pottier. Minimal solutions of linear diophantine systems : bounds and algorithms. In *Proceedings of the Fourth International Conference on Rewriting Techniques and Applications*, pages 162–173, Como, Italy, April 1991.

[25] J. F. Romeuf. A polynomial algorithm for solving systems of two linear diophantine equations. Technical report, Laboratoire d'Informatique de Rouen et LITP, France, 1989.

[26] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[27] J. C. Sogno. Analysis of standard and new algorithms for the integer and linear constraint satisfaction problem. Technical report, INRIA, 1992.

[28] M. Stickel. A unification algorithm for associative-commutative functions. *Journal of the ACM*, 28(3):423–434, 1981.

[29] P Van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, 1989.

[30] P. Van Hentenryck, H. Simonis, and M. Dincbas. Constraint satisfaction using constraint logic programming. *Artificial Intelligence*, 58(1-3):113–159, December 1992.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399