Static Analysis of Linear Congruence Equalities among Variables of a Program

Philippe Granger
Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France
granger@polytechnique.fr

Abstract This paper is dedicated to the presentation of a new static analysis of programs conceived for discovering linear congruence equations satisfied by integer valued variables (or more generally by any set of integer values abstracted from a program). This analysis generalizes both P. Granger's arithmetical congruence analysis and M. Karr's affine equation analysis. An example shows that it can provide valuable results for automatic vectorization.

Key words Abstract Interpretation, Semantic Analysis of Programs, Diophantine Linear Congruence Equation Systems, Compilers, Automatic Vectorization.

1 Introduction

Static analysis of programs (or semantic analysis, or abstract interpretation) consists in determining automatically semantic informations about programs (ie partly describing their behavior at run-time); in practice, it provides valuable results for optimizing and proving them. It has been given a precise and formal framework by P. & R. Cousot in [2, 3, 4]. The way it works consists in interpreting programs on some predefined non-standard (abstract) domain, so that such interpretations always terminate and yield approximate but safe results — namely invariants satisfied by the analyzed program. In practice, the design of a new analysis essentially comes to the choice of both a convenient and relevant abstract domain.

Here we present a new analysis conceived to discover systems of linear congruence equations satisfied by the integer valued variables of programs. More precisely, at a given control point of a given program, for a family $(x_i)_{1 \le i \le n}$ of n integer valued variables, the result of this analysis (ie the discovered invariant at the considered control point) will be a system of equations of the form

$$\sum_{i=1}^{n} \alpha_i x_i \equiv c \; [m],$$

where $\alpha_1, \ldots, \alpha_n, c, m$ are integers and $\equiv$ stands for the arithmetical congruence relation (ie $a \equiv b \; [m]$ if and only if $\exists \, k \in \mathbb{Z}, a = b+km$).

The most evident practical interest of the discovery of such properties concerns variables used as array indices; in particular, they can be especially useful to perform automatic vectorization, as we shall see an example in Section 8 — so that our analyzer can be thought as a part of a vectorizing compiler. This analysis may also be practically relevant to study some integer values abstracted from programs or data-types, such as the length of a list (or of a stack) or the value of a loop counter or the value of a communication counter in parallel programs, as defined by N. Mercouroff [12].

1.1 Semantic analyses of numerical variables

Let us now briefly survey the existing semantic analyses dedicated to the numerical variables of programs. They are divided in two types: the independent and the relational ones, according as they relate the values taken by different variables or not. The ancestor of all is constant propagation, yielding invariants such as $x = a$ (where x is a variable and a belongs to $\mathbb{Z}$ or $\mathbb{Q}$). For a field (e.g. $\mathbb{Q}$), M. Karr has generalized it to a relational version [9]: the corresponding invariants are systems of affine equations, ie of the form

$$\sum_{i=1}^{n} \alpha_i x_i = c .$$

Besides, P. & R. Cousot have extended constant propagation to interval analysis [2], providing invariants such as $x \in [a, b]$, $x \leq a$, etc. Finally, P. Cousot & N. Halbwachs have generalized both preceding analyses by the analysis of linear restraints among variables of a program [5]; the corresponding abstract properties are systems of linear restraints of the type

$$\sum_{i=1}^{n} \alpha_i x_i \leq c .$$

As regards integer valued variables, P. Granger has extended constant propagation to arithmetical congruence analysis [7], with abstract properties of the type $x \equiv c\ [m]$. So the analysis here presented generalizes both arithmetical congruences and M. Karr's systems of affine equations (since the congruence modulo 0 is the equality relation). Let us finally give the different results yielded by all these analyses applied to the following program fragment (where the two ——'s stand for Boolean expressions not taken into consideration):

```
$i := 0$;  $j := 0$;  $k := 0$;
while —— do
    $i := i+4$;
    if —— then $j := j+4$
           else $j := j+12$;
    $k := k+4$;
od;
```

At the control point immediately following the last assignment, constant propagation obviously gives no information, but M. Karr's affine equation analysis yields $i = k$, interval analysis yields $i, j, k \geq 4$, linear restraint analysis returns the system

$$\begin{cases} i \leq j \leq 3i \\ i \geq 4 \\ i = k \end{cases}$$

while congruence analysis obtains $i, j, k \equiv 0$ [4], and finally our relational congruence analysis computes the system

$$\begin{cases} i \equiv j\ [8] \\ i \equiv 0\ [4] \\ i = k \end{cases}$$

1.2 Overview

In Section 2 we shortly recall the notions prerequisite to a good understanding of the following: first the usual design method of a semantic analysis framework as presented in [4], secondly the notion of congruence analysis in an abelian group given in [7]. Section 3 is dedicated to the characterization of the approximate lattice (ie the abstract domain) associated with systems of linear congruence equations: we exhibit two types of representation for its elements. Then in Section 4 we specify algorithms performing the corresponding lattice operations, namely comparison and least upper bound (lub). Section 5 deals with abstract assignment: abstract addition and multiplication and division by a constant are characterized. Section 6 is devoted to solving linear congruence equations and performing corresponding conditionals. The complexity of the analysis is studied in Section 7, then in Section 8 two examples are presented, one of them specially dedicated to automatic vectorization. Section 9 concludes this paper by a number of further issues.

2 Preliminaries

2.1 Design of a semantic analysis framework

First let us briefly recall the main features of semantic analysis of programs as defined by the Cousots [2, 3] and especially the systematic design method of a semantic analysis framework presented in [4], which will be followed hereafter.

The operational semantics of a program P can be classically specified by a relation on a set of states S. With the aim of characterizing the set D of all reachable states (viz the set of all descendants of the entry states), this relation can also be viewed (by a point to point extension) as an isotone operator F on the powerset $\mathcal{P}(S)$ partially ordered by set inclusion such that the set D is equal to the least fixed point (lfp) of F in the complete lattice $\mathcal{P}(S)(\subseteq)$; it is called static or collecting semantics. This characterization of D suggests to compute it in an iterative way, but such an iteration may be infinite: determining D is indeed an undecidable problem. The basic idea of (approximate) semantic analysis of programs is to make this computation always possible by replacing the complete lattice $\mathcal{P}(S)(\subseteq)$ by a simpler, approximate one, and F by an approximate operator on the approximate lattice, in such a way that the corresponding iteration is guaranteed to terminate, yielding an approximate, but safe result (ie a set D_a such that D is included in D_a). This can be done by means of a pair of adjoined functions:

DEFINITION 1 *Let $L(\leq)$ and $M(\leq)$ be two complete lattices and α be a mapping from L to M and γ be a mapping from M to L; (α, γ) is a pair of adjoined functions if and only if*

$$\forall\, x \in L, \forall\, y \in M,\ \ \alpha(x) \leq y \ \Leftrightarrow\ x \leq \gamma(y).$$

In practice, α will be called abstraction function and γ concretization function. The interest of this notion for our purpose immediately stems from the following result:

PROPOSITION 1 *Let F be an isotone operator on $L(\leq)$ and F_a an isotone operator on $M(\leq)$ greater than or equal to $\alpha \circ F \circ \gamma$, then*

$$\text{lfp}(F) \leq \gamma(\text{lfp}(F_a)).$$

So when $L(\leq)$ is $\mathcal{P}(S)(\subseteq)$, the set $\gamma(\text{lfp}(F_a))$ can stand for D_a; moreover, if the complete lattice $M(\leq)$ has been correctly chosen, this set can be automatically computed. For instance, if $M(\leq)$ is finite or satisfies the ascending chain condition (ie any increasing sequence is stationary), then $\text{lfp}(F_a)$ can indeed be calculated as the result of an iterative computation. In practice, the design of a semantic analysis framework therefore amounts to the choice of a suitable approximate complete lattice $M(\leq)$ and the determination of an approximate operator F_a. Moreover, F_a can be synthesized from the approximate operators associated with the operators on $\mathcal{P}(S)(\subseteq)$ standing for the primitive operations in the programming language, and only these ones must be determined in practice.

There exist alternative ways for specifying a semantic analysis framework, using for instance an upper closure operator on the complete lattice $\mathcal{P}(S)(\subseteq)$ or a Moore family of it. Let us simply recall the definition of a Moore family of a complete lattice:

DEFINITION 2 *A Moore family E of a complete lattice $L(\leq)$ is a subset of L completely closed under greatest lower bound (glb), ie $\forall\, S \subseteq E, \wedge S \in E$.*

Hence E contains the greatest element of $L(\leq)$ and is itself a complete lattice (for $\leq$), and

$$(\lambda x. \wedge \{y \in E \,/\, x \leq y\},\ \lambda x.x)$$

is a pair of adjoined functions, so that one can choose equivalently a Moore family of $\mathcal{P}(S)(\subseteq)$ or a pair of adjoined functions. However, once a Moore family has been chosen, its elements must be given a computer representation. We proceed in such a way hereafter, with S equal to $[1, p] \times \mathbb{Z}^n$, corresponding to a program with p control points and n integer valued variables; moreover, so-called partitioning results allow us to restrict S to $\mathbb{Z}^n$, so representing the semantics of the program at a single control point.

After this design stage, the semantic analyzer remains to be actually programmed: this phase essentially amounts to adding the abstract operators (on the approximate lattice) corresponding to the primitive operations of the programming language to a kernel consisting of two modules: one that transforms any program into a system of fixed point equations (to unknowns in the approximate lattice), and one that solves this system iteratively.

2.2 Congruence analysis in an abelian group

Now following the methodology described in the previous paragraph, we have to select a Moore family of the complete lattice $\mathcal{P}(\mathbb{Z}^n)(\subseteq)$ suitable for stating congruence properties relating integer valued variables of programs. With this object, we use the general notion of congruence analysis in an abelian group presented in [7]; precisely, this one amounts to the algebraic definition of a Moore family of the complete lattice $\mathcal{P}(G)(\subseteq)$ (dealing with congruence properties), where G is an additive group (in our special case, G will be $\mathbb{Z}^n$). More details about the following notions are available in any treatise on elementary algebra, for instance [8, 11]; here we simply recall that any subset of G of the form

$$a+H = \{x \in G \,/\, \exists\, h \in H, x = a+h\}$$

where a is an element of G and H is a subgroup of G, is called a coset (of H) in G; $a+H$ is also the class containing a with respect to the congruence relation modulo the subgroup H, and a is said to be a representative of the coset and H the (unique) modulo of the coset. Moreover, for any element b of $a+H$, $a+H = b+H$. Any singleton $\{a\}$ ($= a+\{0\}$) of G is a coset in G, so is G ($= 0+G$) itself. Let us then state (see [7]):

PROPOSITION 2 *The union of the singleton $\{\emptyset\}$ and of the set of all cosets in the abelian group G is a Moore family of the complete lattice $\mathcal{P}(G)(\subseteq)$.*

This Moore family, used as an approximate complete lattice (for set inclusion), is called hereafter congruence lattice of G and denoted $\mathcal{C}(G)(\subseteq)$. The lattice operations in $\mathcal{C}(G)(\subseteq)$, especially comparison (ie set inclusion) and lub denoted $\vee$, have to be characterized, and this can be simplified by the following formulas (proved in [7]). $\mathbb{Z}a$ denotes the subgroup generated by a, ie $\{\ldots, -a-a, -a, 0, a, a+a, \ldots\}$ and H_1+H_2 the subgroup sum of H_1 and H_2, ie $\{x \in G \,/\, \exists\, h_1 \in H_1, h_2 \in H_2, x = h_1+h_2\}$.

PROPOSITION 3 *Let a_1+H_1 and a_2+H_2 be two cosets in G; then:*

i) $\{a_1+H_1 \subseteq a_2+H_2\} \Leftrightarrow \{a_1-a_2 \in H_2\} \wedge \{H_1 \subseteq H_2\}$

ii) $a_1+H_1 \vee a_2+H_2 = a_1+\mathbb{Z}(a_1-a_2)+H_1+H_2$

In other words, these two formulas reduce comparison and lub in $\mathcal{C}(G)(\subseteq)$ to operations on subgroups of G. In [7], the congruence lattice has been studied for G equal to $\mathbb{Z}$; the following is dedicated to the extension of this study to $\mathbb{Z}^n$.

3 Characterization of the congruence lattice

First the cosets in the additive group $\mathbb{Z}^n$, ie the elements of $C(\mathbb{Z}^n)$ different from the empty set, must be characterized and given a convenient representation.

3.1 Normalized representation of cosets

Any coset in $\mathbb{Z}^n$ has form $a+H$, with $a \in \mathbb{Z}^n$ and H a subgroup of $\mathbb{Z}^n$; hence it suffices to know how to represent H. This can be done using a finite generating system of H (ie a finite set of generators of H) by the following proposition [8, 11]:

PROPOSITION 4 *Any subgroup of $\mathbb{Z}^n$ is free with basis of q elements, $0 \leq q \leq n$.*

This means that H can be represented by a tuple $(e_i)_{1\leq i\leq q}$ of q linearly independent points of $\mathbb{Z}^n$ generating it (ie $H = \mathbb{Z}e_1 + \ldots + \mathbb{Z}e_q$). Such a representation is obviously not unique, since H has generally several bases (and often an infinity), but all of them have the same cardinality q, called by definition the *rank* of H; the rank of a coset in $\mathbb{Z}^n$ is by definition the rank of its modulo. Under a practical point of view, Proposition 4 implies that any coset $a+H$ in $\mathbb{Z}^n$ with rank q can be represented by a $n \times (1+q)$ matrix: the coordinates of a form its first row, whereas the q other rows are constituted by the $n \times q$ matrix M of the coordinates of a basis $(e_i)_{1\leq i\leq q}$ of H. Besides, $a+H$ will be also denoted by $a+M\mathbb{Z}^q$, meaning that it is equal to the image of $\mathbb{Z}^q$ by the mapping that associates $a+M\Lambda$ with any $\Lambda \in \mathbb{Z}^q$ (except when $q = 0$, where $M\mathbb{Z}^0$ conventionally stands for the singleton $\{(0, \ldots, 0)\} \subseteq \mathcal{P}(\mathbb{Z}^n)$). For instance in $\mathbb{Z}^3$, we denote equivalently

$$\begin{pmatrix}2\\0\\2\end{pmatrix} + \mathbb{Z}\begin{pmatrix}4\\3\\4\end{pmatrix} + \mathbb{Z}\begin{pmatrix}0\\6\\8\end{pmatrix}$$

or

$$\begin{pmatrix}2\\0\\2\end{pmatrix} + \begin{pmatrix}4 & 0\\3 & 6\\4 & 8\end{pmatrix}\mathbb{Z}^2,$$

both standing for the coset

$$\{(2+4\lambda, 3\lambda+6\mu, 2+4\lambda+8\mu)) \;/\; (\lambda, \mu) \in \mathbb{Z}^2\}.$$

This type of representation of a coset by a representative and a generating system of the modulo is called *parametric* representation; when a basis of the modulo is used, it is called *normalized* representation: we choose it for the actual representation of cosets.

3.2 Diophantine linear congruence equation systems

Here we characterize as expected the elements of $\mathcal{C}(\mathbb{Z}^n)$ in terms of linear congruence equation systems, so getting another type of representation of them, which will turn out to be useful further. First set (E) the Diophantine linear congruence equation

$$\sum_{i=1}^{n} \alpha_i x_i \equiv c \; [m]$$

to the n unknowns $x_1, \ldots, x_n$ ($\alpha_1, \ldots, \alpha_n, c, m \in \mathbb{Z}$) and S the set of all its solutions, and (E_h) the associated homogeneous Diophantine linear congruence equation, ie

$$\sum_{i=1}^{n} \alpha_i x_i \equiv 0 \; [m],$$

and S_h the set of all its solutions.

PROPOSITION 5 *The set $\mathcal{C}(\mathbb{Z}^n)$ is equal to the set of the solution sets of all systems of linear congruence equations.*

PROOF First consider equation (E): the set S of all its solutions is easily shown equal to either the empty set or the sum of a solution $a \in \mathbb{Z}^n$ of (E) and of the set S_h of all solutions of equation (E_h). Since S_h is clearly a subgroup of $\mathbb{Z}^n$, S is either empty or a coset in $\mathbb{Z}^n$, therefore always an element of $\mathcal{C}(\mathbb{Z}^n)$. Finally, the set of all solutions of a system of such equations, equal to the intersection of elements of $\mathcal{C}(\mathbb{Z}^n)$, necessarily belongs to the Moore family $\mathcal{C}(\mathbb{Z}^n)$ of the complete lattice $\mathcal{P}(\mathbb{Z}^n)(\subseteq, \cup, \cap)$.

Conversely, let C be an element of $\mathcal{C}(\mathbb{Z}^n)$: if C is empty, it can be equivalently defined by any linear congruence equation having no solution (eg $2x_1 \equiv 1$ [2]); if C is the singleton $\{a\}$, it is obviously equal to the set of all solutions of the linear equation system $X = a$, where $X = (x_1, \ldots, x_n)$; otherwise $C = a+M\mathbb{Z}^q$, with q greater than 0 and equal to the rank of M, and then for any $X = (x_1, \ldots, x_n) \in \mathbb{Z}^n$,

$$X \in C \Leftrightarrow \exists\, \Lambda \in \mathbb{Z}^q, X = a+M\Lambda.$$

Solving this system to the unknown $\Lambda = (\lambda_1, \ldots, \lambda_q)$ in terms of X used as a vector of parameters yields the equivalent system of n equations

$$\begin{cases} \lambda_i = f_i(x_1, \ldots, x_n) & 1 \leq i \leq q \\ f_j(x_1, \ldots, x_n) = 0 & q+1 \leq j \leq n \end{cases}$$

where any f_i $(1 \leq i \leq n)$ is an affine form on $\mathbb{Q}$ (ie an affine application from $\mathbb{Q}^n$ to $\mathbb{Q}$). Now there exists a convenient positive integer m_i such that $F_i = m_i f_i$ is an affine form with only integer coefficients; then the previous system is equivalent to

$$\begin{cases} m_i\lambda_i = F_i(x_1, \ldots, x_n) & 1 \leq i \leq q \\ F_j(x_1, \ldots, x_n) = 0 & q+1 \leq j \leq n \end{cases}$$

and finally $(x_1, \ldots, x_n)$ belongs to C if and only if

$$\begin{cases} F_i(x_1, \ldots, x_n) \equiv 0\ [m_i] & 1 \le i \le q \\ F_j(x_1, \ldots, x_n) = 0 & q+1 \le j \le n \end{cases}$$

which ends the proof. ♦

This proposition not only achieves the expected characterization of $C(\mathbb{Z}^n)$ (involving in particular that it generalizes both arithmetical congruence analysis [7] and linear equation analysis [9]) but provides another type of representation of its elements, namely by a system of at most n linear congruence equations. This type of representation is clearly suitable for displaying the results of analyses or for entering specifications of programs. Passage algorithms are therefore needed: the proof of Proposition 6 immediately yields one for passing from a normalized representation to an equation system, using classical matrix algorithms [14]. Let us see how it works on the following example: let

$$C = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} \mathbb{Z}^2$$

be a coset in $\mathbb{Z}^3$; (x, y, z) belongs to C if and only if there exists $(\lambda, \mu) \in \mathbb{Z}^2$ such that

$$\begin{cases} x = \lambda \\ y = 1+\lambda+2\mu \\ z = 1+2\lambda \end{cases}$$

Solving this system to the unknowns λ and μ yields

$$\begin{cases} \lambda = x \\ \mu = (-x+y-1)/2 \\ 2x-z+1 = 0 \end{cases}$$

After multiplying the second equation by 2, then eliminating λ and μ, a representation of C by an equation system is eventually reached, viz

$$\begin{cases} x \equiv 0\ [1] \\ -x+y-1 \equiv 0\ [2] \\ 2x-z+1 = 0 \end{cases}$$

the first equation being useless and the second one equivalent to $x+y \equiv 1$ [2]. So this passage algorithm is simply an elimination algorithm; a passage algorithm for the opposite way will be specified below (6.3).

3.3 Ascending chain condition

With the purpose of guaranteeing the systematic termination of the analysis, let us prove that the congruence lattice $C(\mathbb{Z}^n)(\subseteq)$ satisfies the ascending chain condition: let

$$C_0 \subset C_1 \subset C_2 \subset \ldots \subset C_i \subset \ldots$$

be a strictly increasing sequence of elements of $C(\mathbb{Z}^n)$; only C_0 may be equal to the empty set, hence for any $i \geq 1$, C_i is equal to a coset a_i+H_i in $\mathbb{Z}^n$; moreover, since C_i contains C_1, C_i also equals a_1+H_i. Now the sequence deprived of its first element C_0, ie

$$a_1+H_1 \subset a_1+H_2 \subset \ldots \subset a_1+H_i \subset \ldots$$

is obviously isomorphic to the strictly increasing sequence of subgroups of $\mathbb{Z}^n$

$$H_1 \subset H_2 \subset \ldots \subset H_i \subset \ldots$$

which is necessarily finite, because the complete lattice of all subgroups of $\mathbb{Z}^n$ satisfies the ascending chain condition [11]. Hence the initial sequence is also finite (although there is no upper bound of its length), ensuring that the analysis always terminates.

4 Lattice operations

Now that the elements of $C(\mathbb{Z}^n)(\subseteq)$ have been given a convenient representation (using a special symbol for the empty set), algorithms achieving the lattice operations on this representation must be specified. In this section, only comparison (ie set inclusion) and lub in $C(\mathbb{Z}^n)(\subseteq)$ are considered: comparison is needed to stop the analysis, whereas lub is used for abstracting loops. glb, viz set intersection, which is needed only for combining forwards and backwards analyses, is studied further.

4.1 Comparison

Proposition 3 states that a coset a_1+H_1 is included in a coset a_2+H_2 if and only if a_1-a_2 belongs to H_2 and H_1 is included in H_2; H_1 being represented by a basis $(e_i)_{1 \leq i \leq q}$, H_1 is included in H_2 if and only if all vectors e_i, $1 \leq i \leq q$, belong to H_2. So comparison in $C(\mathbb{Z}^n)(\subseteq)$ amounts to testing whether an element of $\mathbb{Z}^n$, say a, belongs to a subgroup of $\mathbb{Z}^n$, say H. Such a test can be immediately achieved as soon as H is given a representation by a system of equations: then it suffices to verify that the coordinates of a satisfy the system. Moreover, this system can be computed using the algorithm described in 3.2. In practice, with the purpose of comparing a_1+H_1 and a_2+H_2, and if a system of equations characterizing a_2+H_2 is known, then the associated homogeneous system characterizes H_2 and can be used directly. For instance, if a_2+H_2 is the coset C used as an example in 3.2, then H_2 is described by the associated homogeneous system

$$\begin{cases} x+y \equiv 0 \ [2] \\ 2x-z = 0 \end{cases}$$

and obviously contains

$$a = \begin{pmatrix} 3 \\ 1 \\ 6 \end{pmatrix} \text{ and } e = \begin{pmatrix} 12 \\ -12 \\ 24 \end{pmatrix};$$

therefore the coset $a+a_2+\mathbb{Z}e$ is included in C.

4.2 Normalization algorithm

By Proposition 3, the lub in $\mathcal{C}(\mathbb{Z}^n)(\sqsubseteq)$ amounts to adding subgroups of $\mathbb{Z}^n$, namely:

$$a_1+H_1 \vee a_2+H_2 = a_1+\mathbb{Z}(a_1-a_2)+H_1+H_2.$$

Hence the resulting coset can be immediately deduced from normalized representations of a_1+H_1 and a_2+H_2: a_1 is a representative of it and its modulo is described by the generating system made up of the vector a_1-a_2 and of the basis of H_1 and of the basis of H_2, unfortunately not free in general. So a normalized representation of this coset can be obtained using an algorithm computing an equivalent but free generating system: this algorithm will consist in applying iteratively the so-called normalization algorithm that takes as entry a tuple of $q+1$ elements of $\mathbb{Z}^n$ generating a subgroup H of rank q, and yields a tuple of q linearly independent vectors of $\mathbb{Z}^n$ generating H (ie a basis of H). This algorithm is a simple generalization of Euclid's algorithm. $\lfloor x \rfloor$ denotes the floor of real x, ie the greatest integer less than or equal to x.

PROPOSITION 6 *Let $(e_i)_{1\leq i\leq q+1}$ be a family of $q+1$ elements of $\mathbb{Z}^n$ generating the subgroup H of $\mathbb{Z}^n$ such that $e_1, \dots, e_q$ are linearly independent and $e_{q+1} = \lambda_1 e_1 + \dots + \lambda_q e_q$, with $\lambda_1, \dots, \lambda_q \in \mathbb{Q}$; let $M = (e_{ij})_{1\leq i\leq n,\ 1\leq j\leq q}$ be the $n \times q$ matrix of the coordinates of the q points $(e_i)_{1\leq i\leq q}$ and ϕ be the function defined from $\{-1, 1\} \times [1, q]$ to $\mathbb{Q}$ by: $\phi(x, j) = x\lambda_j - \lfloor x\lambda_j \rfloor$.*

```
while $\phi \neq 0$ do
    choose $(x, j)$ such that $\phi(x, j) \neq 0$;
    for $i = 1, \dots, n$   $e_{ij} := \phi(x, 1)e_{i1} + \dots + \phi(x, q)e_{iq}$;
    for $i = 1, \dots, q, i \neq j$   $\lambda_i := -\phi(x, i)/\phi(x, j)$;
    $\lambda_j := 1/\phi(x, j)$;
od;
```

This algorithm terminates and overwrites M with the matrix of the coordinates of a basis of H.

PROOF In the following, the point $\lambda_1 e_1 + \dots + \lambda_q e_q$ is also denoted e_{q+1} (consistently with its initial value). Now let us prove by induction on the number of iteration steps of the "while" loop that at its entry point, the family $(e_1, \dots, e_{q+1})$ is a generating system of H:

– This is originally true by definition of H.

– Assume this is true after k iteration steps: the induction is relevant only when $\phi \neq 0$, ie when the set $\{(x, j) \;/\; \phi(x, j) \neq 0\}$ is nonempty — so that one of its elements can be actually chosen. In this case, the point e_j is then overwritten with

$$\sum_{i=1}^{q} \phi(x, i)e_i = xe_{q+1} - \lfloor x\lambda_1 \rfloor e_1 - \dots - \lfloor x\lambda_q \rfloor e_q,$$

ensuring that it belongs to $\mathbb{Z}^n$. The values of the λ_i's are finally modified, corresponding to the (virtual) overwriting of e_{q+1} with

$$-\sum_{i=1}^{j-1}(\phi(x,i)/\phi(x,j))e_i + 1/\phi(x,j)e_j - \sum_{i=j+1}^{q}(\phi(x,i)/\phi(x,j))e_i,$$

which is obviously equal to the value of e_j at the entry of the loop. Finally, the complete body of the "while" loop overwrites $(e_1, \dots, e_{q+1})$ with

$$(e_1, \dots, e_{j-1}, xe_{q+1} - \lfloor x\lambda_1 \rfloor e_1 - \dots - \lfloor x\lambda_q \rfloor e_q, e_{j+1}, \dots, e_q, e_j):$$

both of them clearly generate the same subgroup of $\mathbb{Z}^n$, viz H, and this proves the property for the k+1th iteration step.

The algorithm terminates when ϕ becomes equal to 0, which means that all λ_i's are integer numbers, so that e_{q+1} belongs to the subgroup generated by $(e_1, \dots, e_q)$, which generates therefore H and is necessarily a basis of it. Now let us prove that ϕ cannot be always different from 0. First set K the greatest subgroup of $\mathbb{Z}^n$ included in the subspace of the $\mathbb{Q}$-linear space $\mathbb{Q}^n$ generated by H, and choose a basis B of K: the existence of K is guaranteed by the structure of complete lattice of the set of all subgroups of $\mathbb{Z}^n$ partially ordered by set inclusion (its lub being the sum of subgroups [8]). Then for any iteration step k, define $V_k = |\det(e_1, \dots, e_q)|$, where det denotes the determinant relative to basis B: V_k is obviously a positive integer, since any e_i belongs to K (for H is included in K). Now if at the kth step, ϕ is different from 0, then

$$V_{k+1} = |\det(e_1, \dots, e_{j-1}, \sum_{i=1}^{q}\phi(x,i)e_i, e_{j+1}, \dots, e_q)| = \phi(x,j)V_k,$$

where $\phi(x,j) \in \,]0, 1[$. Hence if ϕ never equals 0, then $(V_k)_{k\in\mathbb{N}}$ is a strictly decreasing sequence of positive integers, which is impossible. ♦

In practice, starting with $M = (e_i)_{1\le i\le q+1}$ involves solving first the system of linear equations

$$e_{q+1} = \lambda_1 e_1 + \dots + \lambda_q e_q$$

to the unknowns $\lambda_1, \dots, \lambda_q \in \mathbb{Q}$ [14]: if it has no solution, $e_1, \dots, e_{q+1}$ are linearly independent, otherwise it has only one solution (provided that $(e_i)_{1\le i\le q}$ is free) and the algorithm can then start. For instance, consider the three elements of $\mathbb{Z}^3$

$$a = \begin{pmatrix} 7 \\ 3 \\ 4 \end{pmatrix}, \ b = \begin{pmatrix} -7 \\ -1 \\ 1 \end{pmatrix}, \ c = \begin{pmatrix} -6 \\ 2 \\ 8 \end{pmatrix}:$$

a and b are obviously linearly independent. First solving the equation system

$$c = \lambda a + \mu b$$

yields the unique solution

$$\lambda = \frac{10}{7}, \ \mu = \frac{16}{7}.$$

Then the algorithm can start with $M = (a, b)$: the first value of ϕ is

$$\phi(1,1) = \frac{3}{7}, \ \phi(1,2) = \frac{2}{7}, \ \phi(-1,1) = \frac{4}{7}, \ \phi(-1,2) = \frac{5}{7},$$

(and ϕ is different from 0); then choosing the couple (1, 2), set

$$c' = \phi(1,1)a + \phi(1,2)b = c-a-2b = \begin{pmatrix}1\\1\\2\end{pmatrix}:$$

M becomes equal to (a, c'), λ to $-3/2$ and μ to $7/2$, so that the expression of b in terms of a and c' is:

$$b = -\frac{3}{2}a + \frac{7}{2}c'.$$

At the following step, ϕ is the constant function equal to $1/2$; now choosing $(1, 1)$, set

$$b' = \frac{1}{2}a + \frac{1}{2}c' = b+2a-3c' = \begin{pmatrix}4\\2\\3\end{pmatrix}:$$

M becomes equal to (b', c'), λ to 2 and μ to -1 (viz $a = 2b'-c'$), hence ϕ equals 0 and the execution terminates. Therefore (b', c') is a basis of the subgroup of $\mathbb{Z}^3$ generated by (a, b, c).

5 Abstract arithmetical operations

The design of the analysis must now be achieved by the characterization of the abstract operations corresponding to the primitive operations of the programming language. First let us recall the definition of this notion [3]: the (best) abstract operator associated to a given operator F on $\mathcal{P}(\mathbb{Z}^n)$ is the operator F' on $\mathcal{C}(\mathbb{Z}^n)$ that associates to any $A \in \mathcal{C}(\mathbb{Z}^n)$ the least element in $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$ containing $F(A)$; any operator F'' on $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$ greater than F' is another suitable abstract operation, but not the best one. This section is devoted to abstract assignment, which amounts to abstract basic arithmetical operations; affine transformation and multiplication and division by a constant are given here their corresponding abstract operator.

5.1 Affine transformation

Here $\mathrm{r}(H)$ denotes the rank of H and $\mathrm{Ker}(u)$ the set $\{x \in \mathbb{Z}^n \,/\, u(x) = (0, \ldots, 0)\}$.

PROPOSITION 7 *Let F be an affine transformation on $\mathbb{Z}^n$, u be its linear part, $a \in \mathbb{Z}^n$, $(e_i)_{1\leq i\leq q}$ be a basis of the subgroup H of $\mathbb{Z}^n$; then*

$$F(a+H) = F(a + \mathbb{Z}e_1 + \ldots + \mathbb{Z}e_q) = F(a) + \mathbb{Z}u(e_1) + \ldots + \mathbb{Z}u(e_q) = F(a)+u(H),$$

the rank of $u(H)$ being equal to $q-\mathrm{r}(\mathrm{Ker}(u) \cap H)$.

This is a quite classical result of linear algebra; it implies in particular that F' coincides with F. In practice, F stands for the effect of an assignment of the type

$$x_1 := \alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n;$$

with $\alpha_0, \ldots, \alpha_n \in \mathbb{Z}$. Hence Proposition 7 specifies how a normalized representation of the coset $a+H$ is transformed by the previous assignment into a parametric representation of the resulting coset. When α_1 is different from 0, the rank of $u(H)$ is q and the result is therefore normalized; when $\alpha_1 = 0$, the rank of $u(H)$ is equal to q or $q-1$ and the result is not necessarily normalized. In order to check that it is, the linear system

$$\lambda_1 u(e_1) + \ldots + \lambda_q u(e_q) = (0, \ldots, 0)$$

to the q unknowns $\lambda_1, \ldots, \lambda_q \in \mathbb{Q}$ must be solved [14]: if it has only one solution, viz all λ_i's equal to 0, then $(u(e_i))_{1\leq i\leq q}$ is a basis of the subgroup $u(H)$ of $\mathbb{Z}^n$, otherwise its rank is $q-1$ and the discovered relationship among the vectors of $(u(e_i))_{1\leq i\leq q}$ can initiate the normalization algorithm (4.2) which yields a basis of $u(H)$ when applied to this family. For instance in $\mathbb{Z}^3$, the coset

$$\begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 & 0 \\ 3 & 6 \\ 4 & 8 \end{pmatrix} \mathbb{Z}^2$$

(standing for the values of variables x, y, z) is changed by the assignment

$$x := 3+4y-z;$$

into the coset specified by the parametric representation

$$\begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} + \begin{pmatrix} 8 & 16 \\ 3 & 6 \\ 4 & 8 \end{pmatrix} \mathbb{Z}^2$$

which can be normalized into

$$\begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} + \begin{pmatrix} 8 \\ 3 \\ 4 \end{pmatrix} \mathbb{Z}^2.$$

5.2 Multiplication

ε_i denotes the vector of $\mathbb{Z}^n$ having all its coordinates equal to 0 but the ith equal to 1, pr_i the projection from $\mathbb{Z}^n$ onto its ith component. $\gcd(A)$ denotes the greatest common divisor of the set A of non-negative integers [7, 13]; it can be extended to any subset B of $\mathbb{Z}$ by $\gcd(B) = \gcd(\{|x| \in \mathbb{N} \,/\, x \in B\})$.

PROPOSITION 8 *Let $a = (0, a_2, \ldots, a_n) \in \{0\} \times \mathbb{Z}^{n-1}$ ($n \geq 2$), H be a subgroup of $\mathbb{Z}^n$ included in $\{0\} \times \mathbb{Z}^{n-1}$, $(e_i)_{1\leq i\leq q}$ be a basis of H, with $e_i = (0, e_{2i}, \ldots, e_{ni})$ ($1 \leq i \leq q$), k and $m \in [2, n]$, F be the operator on $\mathbb{Z}^n$ defined by*

$$F(x_1, x_2, \ldots, x_n) = (x_k x_m, x_2, \ldots, x_n)$$

Then the abstract image $F'(a+H)$ is equal to the coset $a'+H'$, where $a' = a+a_k a_m \varepsilon_1$ and H' is the subgroup of $\mathbb{Z}^n$ generated by the family $(e'_i)_{1\leq i\leq q+1}$ defined by

$$e'_i = e_i + (a_k e_{mi} + e_{ki} a_m + e_{ki} e_{mi})\varepsilon_1 \quad (1 \leq i \leq q)$$

$$e'_{q+1} = \gcd((e_{ki}e_{mj} + e_{kj}e_{mi})_{1\leq i\leq j\leq q})\varepsilon_1$$

Moreover, $(e'_i)_{1\leq i\leq q+1}$ (deprived from e'_{q+1} when $e'_{q+1} = (0, \ldots, 0)$) is a basis of H'.

PROOF For any couple $(i, j) \in [1, q]^2$ such that $i \leq j$, set $e''_{ij} = (e_{ki}e_{mj}+e_{kj}e_{mi})\varepsilon_1$: the family $(e''_{ij})_{1\leq i\leq j\leq q}$ generates the same subgroup of $\mathbb{Z}^n$ as e'_{q+1}, by an elementary result of number theory [7, 13]; hence H' is generated by the union of $(e'_i)_{1\leq i\leq q}$ and of this family. Besides, set $M = (e_{ji})_{1\leq j\leq n,\ 1\leq i\leq q}$ the $n \times q$ matrix of the coordinates of the basis $(e_i)_{1\leq i\leq q}$ of H; then $H = M\mathbb{Z}^q$ and being given $(\lambda_1, \ldots, \lambda_q) \in \mathbb{Z}^q$,

$$\mathrm{pr}_j(F(a+M(\lambda_1, \ldots, \lambda_q))) = a_j + \sum_{i=1}^{q}\lambda_i e_{ji} \qquad (2 \leq j \leq n)$$

whereas on the first component

$$\mathrm{pr}_1(F(a+M(\lambda_1, \ldots, \lambda_q))) = (a_k + \sum_{i=1}^{q}\lambda_i e_{ki})(a_m + \sum_{i=1}^{q}\lambda_i e_{mi})$$

$$= a_k a_m + \sum_{i=1}^{q}\lambda_i(a_k e_{mi}+e_{ki}a_m) + \sum_{i=1}^{q}\sum_{j=1}^{q}\lambda_i\lambda_j e_{ki}e_{mj}$$

$$= a_k a_m + \sum_{i=1}^{q}\lambda_i(a_k e_{mi}+e_{ki}a_m) + \sum_{i=1}^{q}\lambda_i^2 e_{ki}e_{mi} + \sum_{i=1}^{q}\sum_{i<j}\lambda_i\lambda_j(e_{ki}e_{mj}+e_{kj}e_{mi})$$

$$= a_k a_m + \sum_{i=1}^{q}\lambda_i(a_k e_{mi}+e_{ki}a_m+e_{ki}e_{mi}) + \sum_{i=1}^{q}\lambda_i(\lambda_i-1)e_{ki}e_{mi}$$

$$+ \sum_{i=1}^{q}\sum_{i<j}\lambda_i\lambda_j(e_{ki}e_{mj}+e_{kj}e_{mi}),$$

which finally implies that

$$F(a+M(\lambda_1, \ldots, \lambda_q)) = a' + \sum_{i=1}^{q}\lambda_i e'_i + \sum_{i=1}^{q}(\lambda_i(\lambda_i-1)/2)e''_{ii} + \sum_{i=1}^{q}\sum_{i<j}\lambda_i\lambda_j e''_{ij}.$$

Hence $F(a+H)$ is included in $a'+H'$ (since $\lambda_i(\lambda_i-1)/2 \in \mathbb{Z}$), so is the abstract image $F'(a+H)$. Conversely $F(a+H)$ contains

$$F(a) = a', \ F(a+M\varepsilon'_i) = a'+e'_i, \ F(a+M(\varepsilon'_i+\varepsilon'_j)) = a'+e'_i+e'_j+e''_{ij}$$

$(1 \leq i \leq j \leq q)$, so that $a'+H'$ must be included in $F'(a+H)$ by Proposition 3. The final result of linear independence is obvious. ♦

This proposition remains true when $q = 0$; it specifies both the abstract product and the abstract square (for $k = m$). In practice, the variable x_1 of Proposition 8 does not stand for an actual variable, but for an auxiliary variable created just before the assignment in order to store temporarily the product and conventionally initialized to 0, then immediately deleted. For instance, the assignment

$$x := x*y;$$

is simulated by the sequence of assignments

$$u := x*y; \ x := u;$$

where u is a temporary auxiliary variable. By Proposition 8 the first assignment changes the initial coset in $\mathbb{Z}^3$ given in the example of 5.1 into the coset in $\mathbb{Z}^4$

$$\begin{pmatrix}2\\0\\2\\0\end{pmatrix}+\begin{pmatrix}4&0&0\\3&6&0\\4&8&0\\18&12&24\end{pmatrix}\mathbb{Z}^3$$

where u is placed in the fourth position; then the second assignment amounts to a linear transformation and can be achieved by 5.1. In particular, a normalization stage may be required: one easily proves that the variable u can be safely deleted before this stage. So the previous coset is transformed into the coset in $\mathbb{Z}^3$

$$\begin{pmatrix}0\\0\\2\end{pmatrix}+\begin{pmatrix}18&12&24\\3&6&0\\4&8&0\end{pmatrix}\mathbb{Z}^3$$

finally normalized into

$$\begin{pmatrix}0\\0\\2\end{pmatrix}+\begin{pmatrix}18&24\\3&0\\4&0\end{pmatrix}\mathbb{Z}^2.$$

Moreover, any assignment of a polynomial expression can be simulated by a sequence of assignments of linear expressions or simple products; hence it can be treated by 5.1 and 5.2 (by creating as many variables as necessary).

5.3 Division by a constant

Here we partially specify the abstract operator associated to the division by a constant. Precisely, under the assumptions of Proposition 8, let $d \in \mathbb{N}^*$ and F_d and F_m be the operators on $\mathbb{Z}^n$ respectively defined by

$$F_d(x_1, x_2, \ldots, x_n) = (x_k \text{ div } d, x_2, \ldots, x_n)$$

and by

$$F_m(x_1, x_2, \ldots, x_n) = (x_k \text{ mod } d, x_2, \ldots, x_n),$$

with div and mod defined as in PASCAL, ie for any $x \in \mathbb{Z}$,

$$x \text{ div } d = (\text{if } x \geq 0 \text{ then } x \text{ div}_e\, d \text{ else } -(-x \text{ div}_e\, d))$$

and

$$x \text{ mod } d = (\text{if } x \geq 0 \text{ then } x \text{ mod}_e\, d \text{ else } -(-x \text{ mod}_e\, d)),$$

where div_e and mod_e respectively stand for the quotient and the remainder of the euclidian division, ie verifying for any $a \in \mathbb{Z}$ and $b \in \mathbb{N}^*$

$$a = b(a \text{ div}_e\, b)+(a \text{ mod}_e\, b) \text{ and } 0 \leq a \text{ mod}_e\, b \leq b-1.$$

Now the abstract images $F_d'(a+H)$ and $F_m'(a+H)$ depend on the values taken by the variable x_k in the coset $a+H$. If $pr_k(a+H)$ is a singleton, then the result is immediate; otherwise if $pr_k(H)$ is included in $d\mathbb{Z}$, ie all e_{ki}'s multiple of d, one can then prove that

$$F_d'(a+H) = (\text{if } d \mid a_k \text{ then } F_d(a+H) \text{ else } a+H+\mathbb{Z}\varepsilon_1)$$

where

$$F_d(a+H) = a+(a_k \operatorname{div} d)\varepsilon_1 + \sum_{i=1}^{q} \mathbb{Z}(e_i+(e_{ki} \operatorname{div} d)\varepsilon_1)$$

(the family $(e_i+(e_{ki} \operatorname{div} d)\varepsilon_1)_{1\leq i\leq q}$ being obviously free), and that

$$F_m'(a+H) = (\text{if } d \mid a_k \text{ then } a+H \text{ else } a+H+\mathbb{Z}d\varepsilon_1).$$

In all cases, if x_1 is a temporary auxiliary variable, at most one normalization will be finally necessary. At last, if d does not divide all e_{ki}'s, a partial result can then be obtained as regards the abstract image $F_m'(a+H)$, which can be easily shown included in the coset $a+H+\mathbb{Z}\varepsilon_1 \cap S$, where S is the set of all solutions of the linear congruence equation $x_1 \equiv x_k$ $[d]$; hence this coset is a safe approximation of $F_m'(a+H)$. Section 6 deals with the general problem of determining a normalized representation of it.

6 Diophantine linear congruence equations

This section is devoted to the characterization of the abstract operator on $\mathbb{Z}^n$ associated with a linear congruence equality test. Precisely, being given a coset $a+H$ in $\mathbb{Z}^n$ and the linear congruence equation (E) defined in 3.2, we must determine the least element in $C(\mathbb{Z}^n)(\subseteq)$ containing $S \cap a+H$, namely $S \cap a+H$ itself, since this set is a coset by Propositions 2 and 5. So we only have to find a normalized representation of $S \cap a+H$ (if not empty).

6.1 General solution of a linear congruence equation

First let us show how to solve equation (E) in $\mathbb{Z}^n$: here we specify an algorithm yielding a normalized representation of S. As seen before (in the proof of Proposition 5), S is equal to $a+S_h$, where a is any solution of (E) and S_h the set of all solutions of (E_h); with the purpose of first determining such an a, let us state:

PROPOSITION 9 *Let $(a_1, \ldots, a_n)$ be an element of $\mathbb{Z}^n$ such that for any $j \in [1, n]$, a_j is a solution of the linear congruence equation $(E_j(a_1, \ldots, a_{j-1}))$*

$$\alpha_j x \equiv c-\sum_{i=1}^{j-1}\alpha_i a_i \;\; [\gcd(\alpha_{j+1}, \ldots, \alpha_n, m)]$$

to the unknown x. Then $(a_1, \ldots, a_n)$ is a solution of (E); moreover, S is empty if and only if (E_1) has no solution; otherwise all the n preceding equations have a solution.

This proposition can be proved quite easily; it immediately specifies an algorithm which computes a solution of (E) by solving successively n Diophantine linear equations to one unknown. Hence another algorithm dedicated to this work is needed: Euclid's algorithm [7, 13] is classically used; here a version of it suitable for our problem is given (as a matter of convention we set $c_1/0 = 0$):

PROPOSITION 10 *Let α, c, $m \in \mathbb{Z}$ and (E) be the Diophantine linear congruence equation: $\alpha x \equiv c$ $[m]$ (to the unknown x); let α_1, α_2, α_3, c_1, c_2, c_3 be six integer valued variables.*

```
$(\alpha_1, c_1) := (\alpha, c)$; $(\alpha_2, c_2) := (|m|, 0)$;
while $\alpha_2 \neq 0$ do
    $(\alpha_3, c_3) := (\alpha_2, c_2)$;
    $(\alpha_2, c_2) := (\alpha_1 \bmod_e \alpha_3, (c_1 - c_3(\alpha_1 \operatorname{div}_e \alpha_3)) \bmod_e |m|)$;
    $(\alpha_1, c_1) := (\alpha_3, c_3)$;
od;
```

This algorithm terminates with the final values of α_1 and c_1 satisfying:

i) (E) has a solution if and only if α_1 is a divisor of c.

ii) In this case, c_1/α_1 is a solution of (E).

So the expected algorithm can be deduced from both these propositions. Let us see an example how it works: consider in $\mathbb{Z}^3$ the equation

$$4x+5y-5z \equiv 3 \ [10];$$

first the equation (E_1) defined in Proposition 9 is determined, namely

$$4x \equiv 3 \ [5].$$

2 is a solution of (E_1) (obtained by Proposition 10); then equation $(E_2(2))$, ie

$$5y \equiv -5 \ [5]$$

(to the unknown y) must be solved: Euclid's algorithm yields solution 0. Then this algorithm applied to the third equation $(E_3(2, 0))$, ie

$$-5z \equiv -5 \ [10]$$

(to the unknown z) yields solution 1; hence (2, 0, 1) is a solution of the initial equation.

The second part of our work consists in determining a basis of S_h (which is a subgroup of $\mathbb{Z}^n$ by Proposition 5); it can be immediately deduced from the following proposition which is a corollary of a result due to Heger (see [6], page 82):

PROPOSITION 11 *Assuming now α_n different from 0, let $(e_i)_{1 \leq i \leq n}$ be a family of elements of $\mathbb{Z}^n$, with $e_i = (e_{1i}, \dots, e_{ni})$, $1 \leq i \leq n$, such that for any $i, j \in [1, n]$,*

$$\{ i<j \} \Rightarrow \{ e_{ij} = 0 \}, \qquad e_{ii} = \frac{\gcd(\alpha_{i+1}, \dots, \alpha_n, m)}{\gcd(\alpha_i, \dots, \alpha_n, m)},$$

$$\sum_{i=j+1}^{n} \alpha_i e_{ij} \equiv -\alpha_j e_{jj} \ [m].$$

Then such a family always exists and either $(e_i)_{1 \leq i \leq n-1}$ when $m = 0$, or $(e_i)_{1 \leq i \leq n}$ when $m \neq 0$, is a basis of S_h.

Proposition 11 provides a simple way for determining a basis of S_h: set M the $n \times n$ matrix of the coordinates of the family $(e_i)_{1 \le i \le n}$. M is therefore a lower triangular matrix; its diagonal elements are given by a simple formula, whereas its elements below the diagonal can be easily computed as solutions of solvable linear congruence equations, which can be achieved by using the preceding algorithm (deduced from Propositions 9 and 10). Precisely, for any $j \in [1, n-1]$, $(e_{j+1j}, \ldots, e_{nj})$ is a solution of the equation

$$\sum_{i=j+1}^{n} \alpha_i x_i \equiv -\alpha_j e_{jj} \, [m]$$

to the unknown $(x_{j+1}, \ldots, x_n) \in \mathbb{Z}^{n-j}$. For instance, let us determine a basis (e_1, e_2, e_3) of the set of all solutions of the homogeneous equation

$$4x+5y-5z \equiv 0 \, [10].$$

First the three diagonal elements are $e_{11} = 5$, $e_{22} = 1$, $e_{33} = 2$. Then the couple (e_{21}, e_{31}) is computed as a solution of the equation

$$5y-5z \equiv -20 \, [10]$$

eg (0, 0), and e_{32} as a solution of

$$-5z \equiv -5 \, [10]$$

eg 1. Now combining this result with the preceding one, a normalized representation of the coset characterized by the equation

$$4x+5y-5z \equiv 3 \, [10]$$

is finally obtained, namely

$$\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix} \mathbb{Z}^3.$$

6.2 Abstract linear test

Let us now specify the abstract operator corresponding to linear congruence equality test, which simply amounts to solving equation (E) in the coset $a+H$, ie to determining a normalized representation of the coset $S \cap a+H$. Here A denotes $(\alpha_1, \ldots, \alpha_n)$ considered as a line-vector, whereas all the other vectors, ie $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_q)$, a, b are considered as row-vectors; hence equation (E) can also be written

$$AX \equiv c \, [m].$$

H is represented by $M\mathbb{Z}^q$, where M is the $n \times q$ matrix of the coordinates of a basis of H, assuming moreover $q \ge 1$ (the case $q = 0$ being obvious).

PROPOSITION 12 *Let S' be the set of all solutions in $\mathbb{Z}^q$ of the linear congruence equation (E') to the unknown $Y = (y_1, \ldots, y_q)$*

$$AMY \equiv c-Aa \, [m].$$

Then $S \cap a+M\mathbb{Z}^q$ is empty if and only if S' is empty; otherwise $S' = b+N\mathbb{Z}^r$, with $b \in \mathbb{Z}^q$ and $r \in \mathbb{N}$ and N a $q \times r$ matrix of integers of rank r, and

$$S \cap a+M\mathbb{Z}^q = a+Mb+MN\mathbb{Z}^r$$

the rank of MN being r.

PROOF $X \in \mathbb{Z}^n$ belongs to the set $S \cap a+M\mathbb{Z}^q$ if and only if

$$AX \equiv c\,[m] \ \wedge \ \exists\, \Lambda \in \mathbb{Z}^q,\ X = a+M\Lambda$$

if and only if

$$\exists\, \Lambda \in \mathbb{Z}^q,\ AM\Lambda \equiv c-Aa\,[m] \ \wedge \ X = a+M\Lambda$$

if and only if

$$\exists\, \Lambda \in S',\ X = a+M\Lambda$$

if and only if

$$X \in a+MS'.$$

The result about the rank is classical [11]. ♦

So this proposition provides an algorithm computing a normalized representation of $S \cap a+M\mathbb{Z}^q$, which consists in determining and solving the transformed equation (E') (using the algorithm presented in 6.1) then finally calculating $a+Mb$ and MN. For instance, with the object of solving the equation

$$x-y+2z-5t \equiv 0\,[10]$$

to the four (so ordered) unknowns x, y, z, t in

$$\begin{pmatrix}0\\0\\1\\1\end{pmatrix}+\begin{pmatrix}3&2&1\\1&0&1\\1&4&0\\0&1&1\end{pmatrix}\mathbb{Z}^3,$$

first we determine the transformed equation (E') to the unknown (λ, μ, ν) in $\mathbb{Z}^3$, ie

$$4\lambda+5\mu-5\nu \equiv 3\,[10].$$

It suffices then to solve it (which has already be done in 6.1) to obtain a normalized representation of the solution coset $S \cap a+H$, ie

$$\begin{pmatrix}0\\0\\1\\1\end{pmatrix}+\begin{pmatrix}3&2&1\\1&0&1\\1&4&0\\0&1&1\end{pmatrix}\cdot\begin{pmatrix}2\\0\\1\end{pmatrix}+\begin{pmatrix}3&2&1\\1&0&1\\1&4&0\\0&1&1\end{pmatrix}\cdot\begin{pmatrix}5&0&0\\0&1&0\\0&1&2\end{pmatrix}\mathbb{Z}^3 = \begin{pmatrix}7\\3\\3\\2\end{pmatrix}+\begin{pmatrix}15&3&2\\5&1&2\\5&4&0\\0&2&2\end{pmatrix}\mathbb{Z}^3.$$

6.3 Derived algorithms

An algorithm achieving the intersection of cosets in $\mathbb{Z}^n$ can be immediately deduced from the previous algorithm (6.2) used iteratively. Starting with two cosets C_1 and C_2, the first one characterized by a normalized representation, the second one by a system of linear congruence equations (possibly deduced from 3.2), solving the first equation in C_1 (by the previous algorithm), then the second one in the normalized result, and so on up to the last equation, finally yields a normalized form of the coset $C_1 \cap C_2$.

Moreover, this algorithm specialized by setting C_1 equal to $\mathbb{Z}^n$ becomes a passage algorithm from a system of linear congruence equations (ie C_2) to a normalized parametric representation of the coset it defines: this is the converse passage algorithm corresponding to the one specified in 3.2.

7 Complexity of the analysis

The time complexity of the analysis essentially depends upon two factors, namely the maximal complexity of abstract primitive operations (including lattice operations) and the total number of iteration steps before convergence, which amounts to the length of strictly ascending chains in $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$. Now this length is not bounded: by example in $\mathcal{C}(\mathbb{Z})(\subseteq)$, for any $k \in \mathbb{N}$, there exists a strictly increasing chain with length k, eg

$$2^k\mathbb{Z} \subset 2^{k-1}\mathbb{Z} \subset \dots \subset 2^0\mathbb{Z}.$$

We then need an additional assumption in order to characterize the actual complexity of the analysis: we shall suppose that the modulo of any coset occuring in the analysis has a basis of vectors with all coordinates bounded by some integer d. In other words, there exists $d \in \mathbb{N}$ such that for any $a+H$ occuring in the analysis, there exists a basis $(e_i)_{1\leq i\leq q}$ of H such that for any $i \in [1, q]$, $\|e_i\| \leq d$ (where $\|(x_1, \dots, x_n)\| = \max(|x_1|, \dots, |x_n|)$).

In PASCAL, integer valued variables belong to the interval [–maxint–1, maxint], so that the only relevant cosets, ie describing values that can be actually taken by variables, are those verifying the assumption with d = 2maxint+1 (then justifying it). Yet we must notice that in practice this "natural" assumption is not guaranteed to be always satisfied, although it can be enforced by using a so-called widening operator [2, 3].

Under this assumption one can show that not only the length of strictly ascending chains in $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$, but also the number of iteration steps of the normalization algorithm are bounded. Going back indeed to Proposition 6, consider the integer numbers V_k defined in the proof: if at the kth step ϕ is different from 0, then $V_{k+1} = \phi(x,j)V_k$; suppose we have chosen an x in $\{1, -1\}$ that minimizes ϕ, then $\phi(x,j)$ is less than or equal to 1/2 (since $\phi(-x,j)+\phi(x,j) = 1$). Now if ϕ becomes the zero function at the mth step, this involves that

$$V_m \leq \frac{V_0}{2^m}$$

which directly implies that

$$m \leq \log_2 V_0.$$

But V_0 can be easily proved less than or equal to $(d\sqrt{q})^q$, so that finally:

$$m \leq q(\frac{1}{2}\log_2 q + \log_2 d).$$

	init.	1st step	2nd step	3rd step
{1:}	$\varnothing$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$
{2:}	$\varnothing$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}+\begin{pmatrix}1\\3\\2\end{pmatrix}\mathbb{Z}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}+\begin{pmatrix}1&1\\3&5\\2&2\end{pmatrix}\mathbb{Z}^2$
{3:}	$\varnothing$	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\\1\end{pmatrix}+\begin{pmatrix}1\\3\\2\end{pmatrix}\mathbb{Z}$	$\begin{pmatrix}1\\1\\1\end{pmatrix}+\begin{pmatrix}1&1\\3&5\\2&2\end{pmatrix}\mathbb{Z}^2$
{4:}	$\varnothing$	$\begin{pmatrix}1\\1\\3\end{pmatrix}$	$\begin{pmatrix}1\\1\\3\end{pmatrix}+\begin{pmatrix}1\\3\\2\end{pmatrix}\mathbb{Z}$	$\begin{pmatrix}1\\1\\3\end{pmatrix}+\begin{pmatrix}1&1\\3&5\\2&2\end{pmatrix}\mathbb{Z}^2$
{5:}	$\varnothing$	$\begin{pmatrix}1\\4\\3\end{pmatrix}$	$\begin{pmatrix}1\\4\\3\end{pmatrix}+\begin{pmatrix}1\\5\\2\end{pmatrix}\mathbb{Z}$	$\begin{pmatrix}1\\4\\3\end{pmatrix}+\begin{pmatrix}1&1\\5&7\\2&2\end{pmatrix}\mathbb{Z}^2$
{6:}	$\varnothing$	$\begin{pmatrix}0\\1\\1\end{pmatrix}+\begin{pmatrix}1\\3\\2\end{pmatrix}\mathbb{Z}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}+\begin{pmatrix}1&1\\3&5\\2&2\end{pmatrix}\mathbb{Z}^2$	$\begin{pmatrix}0\\1\\1\end{pmatrix}+\begin{pmatrix}1&1\\3&5\\2&2\end{pmatrix}\mathbb{Z}^2$

Table I

In a very similar way, one shows that the length of any strictly increasing chain of cosets in $\mathbb{Z}^n$ with rank q cannot exceed the previous value, and therefore that the length of any strictly increasing chain in $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$ is bounded by

$$1 + n + \frac{n(n+1)}{2}\left(\frac{1}{2}\log_2 n + \log_2 d\right).$$

After the first $O(n^3)$ multiplications (or divisions) needed to solve the initial linear system, the normalization algorithm then requires $O(n^2)$ multiplications at each one of the $O(n\log_2 n)$ iteration steps, hence $O(n^3\log_2 n)$ multiplications for the whole ($O(u_n)$ denoting any sequence v_n such that $|v_n| \leq \alpha|u_n|$ for some positive number α and n large). The lub in $\mathcal{C}(\mathbb{Z}^n)(\subseteq)$, corresponding to at most n+1 successive normalizations, then requires $O(n^4\log_2 n)$ multiplications; similarly, $O(n^3\log_2 n)$ are needed for any abstract arithmetical operations (Section 5). Finally, one verifies easily that $O(n^3)$ are required by comparison (4.1) and abstract linear test (6.2), and $O(n^4)$ by the algorithms presented in 6.3. Now we can conclude that the total number of multiplications (or divisions) executed for the analysis of a program will not exceed

$$\alpha p n^6 (\log_2 n)^2$$

for some positive number α (depending on d) and n large, p representing the size of the analyzed program. This is a rather satisfactory result when compared to the inherently exponential complexity of linear restraint analysis [5]; moreover, it is but an upper bound of the worst case, and the average complexity of the analysis is definitely far better. Finally, the preceding results suggest that the complexity might be improved by replacing the most expensive algorithms (essentially interesting for their adaptability), namely lub and intersection, by more efficient ones (see for instance [6, 10]).

8 Two examples

Let us now illustrate with two examples some concrete results provided by the analysis.

8.1 Iteration sequence

The first example consists in the simple forwards analysis of a procedure which computes (in variable x) the integer square root of variable n:

```
        x:=0; y:=1; z:=1;
{1:}
        while y ≤ n do
{2:}
                x:=x+1;
{3:}
                z:=z+2;
{4:}
                y:=y+z;
{5:}
        od;
{6:}
```

The analysis converges in three iteration steps detailed in Table I. We can immediately notice that once a representative has been computed (here at the first iteration step), it is always uselessly recomputed at the following steps. This is a quite general fact; in practice this overhead can be easily avoided. We can also notice that in one step the moduli are not necessarily deeply changed: for instance at control point {6:}, the second iteration step corresponds to the computation of the lub

$$\begin{pmatrix}0\\1\\1\end{pmatrix} \vee \begin{pmatrix}1\\4\\3\end{pmatrix} + \begin{pmatrix}1\\5\\2\end{pmatrix}\mathbb{Z}$$

and the third iteration step to the computation of

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \vee \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 5 & 7 \\ 2 & 2 \end{pmatrix} \mathbb{Z}^2,$$

so that only the part of the calculus involved by vector (1, 7, 2) is new: this may be taken into account in order to improve the complexity of the analysis. Finally the discovered invariant at the control point {6:} is the equation system

$$\begin{cases} 2x-z+1 = 0 \\ x+y \equiv 1 \ [2] \end{cases}$$

to be compared with the exact invariant relating x, y, z:

$$(x, y, z) \in \{(m, (m+1)^2, 2m+1 \mathbin{/} m \in \mathbb{N}\}.$$

8.2 Automatic vectorization

Next example shows the interest of the analysis for automatic vectorization; consider the following procedure classically used to solve Laplace's equation by numerical means:

```
procedure Semi_Iteration(r: integer);
for i := 1 to N do
    for j := 1 to N do
        if ((i+j) mod 2 = r) then
            A[i, j] := (A[i-1, j]+A[i+1, j]+A[i, j-1]+A[i, j+1]) / 4.0;
    od;
od;
```

It is called alternately with r equal to 0 or 1. The analysis of this procedure called with 0 applied to the values of the two indices x and y of the array A allows to discover that they satisfy the equation $x+y \equiv 0$ [2] on the left-hand side of the assignment and the equation $x+y \equiv 1$ [2] on the right-hand side. Since the intersection of these two cosets is empty, the assignment does not depend upon itself (see [1]). The result is similar when the procedure is called with 1, finally implying that the loops can be licitly converted to vector form (and all of this can be done automatically).

9 Conclusion

We have presented a new semantic analysis framework conceived for discovering linear congruence equations satisfied by integer valued variables of programs. This framework is likely to be improved, for instance by using more efficient algorithms as suggested in Section 7, or by representing cosets differently (eg by using equation systems, particular representations of matrices, etc.). Besides, we have shown an example how it can be useful for automatic vectorization: yet better informations could be obtained with a more suitable framework derived from ours (eg by using a fixed modulo and analyzing the representatives). N. Mercouroff has suggested its interest for static analysis of communications in parallel programs through the analysis of a communication counter [12]. Similarly, it can be used to study other integer abstract values (length of a list, depth of a tree, etc.), and for this purpose it must be adapted to natural numbers: for instance it can be extended into a more general framework obtained by combining it [4] with other analyses (intervals, linear restraints, etc.).

References

[1] R. ALLEN & K. KENNEDY, Automatic translation of FORTRAN program to vector form, *ACM Transactions on Programming Languages and Systems* 9(4), 491-542, 1987.

[2] P. & R. COUSOT, Static determination of dynamic properties of programs, *Proc. 2nd Int. Symp. on Programming*, Dunod, Paris, 106-130, 1976.

[3] P. & R. COUSOT, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages*, Los Angeles, Calif., 238-252, 1977.

[4] P. & R. COUSOT, Systematic design of program analysis frameworks, *Conf. Rec. of the 6th ACM Symp. on Principles of Programming Languages*, San Antonio, Texas, 269-282, 1979.

[5] P. COUSOT & N. HALBWACHS, Automatic discovery of linear restraints among variables of a program, *Conf. Rec. of the 5th ACM Symp. on Principles of Programming Languages*, Tucson, Ariz., 84-97, 1978.

[6] L.E. DICKSON, *History of the theory of numbers*, vol. 2, Carnegie Institution, Washington, 1919.

[7] P. GRANGER, Static analysis of arithmetical congruences, *Int. Journal of Computer Mathematics* **30**, 165-190, 1989.

[8] N. JACOBSON, *Basic Algebra I*, W.H. Freeman and Company, 1985.

[9] M. KARR, Affine relationships among variables of a program, *Acta Informatica* **6**, 133-151, 1976.

[10] D.E. KNUTH, *The art of computer programming*, 2nd ed., vol. 2, 326-327, Addison-Wesley, 1981.

[11] S. LANG, *Algebra*, Addison-Wesley, 1965.

[12] N. MERCOUROFF, *Analyse sémantique des communications entre processus de programmes parallèles*, Thèse de l'Ecole Polytechnique, 1990.

[13] H.M. STARK, *An introduction to number theory*, Markham Publishing Company, 1970.

[14] G.W. STEWART, *Introduction to matrix computations*, Academic Press, 1973.