Détection des fonctions de rang linéaires à terme

Anthony Alezan[1] **Roberto Bagnara**[2] **Fred Mesnard**[1] **Étienne Payet**[1]

[1] LIM/ERIMIA, université de la Réunion, France

[2] BUGSENG & Dipartimento di Matematica e Informatica, Università di Parma, Italie

{anthony.alezan, fred, epayet}@univ-reunion.fr bagnara@cs.unipr.it

Résumé

La terminaison des programmes est un sujet actif de recherche en informatique. Ces dernières années ont vu l'apparition d'analyseurs de terminaison performants pour des langages comme C ou Java où l'emploi des techniques et outils de la programmation par contraintes est omniprésent. Dans cet article, nous rappelons un algorithme particulier basé sur l'emploi du lemme de Farkas pour le calcul de fonctions de rang linéaires garantissant la terminaison d'une certaine classe de boucles. Puis nous présentons une extension de cette méthode pour la découverte de fonctions linéaires qui deviennent des fonctions de rang linéaire à terme, c.-à-d. après un certain nombre de passages dans la boucle. Nous montrons la correction et la complétude d'un algorithme polynomial pour ce problème.

Abstract

Program termination is a hot research topic in program analysis. The last few years have witnessed the development of termination analyzers for programming languages such as C and Java with remarkable precision and performance. These systems are largely based on techniques and tools coming from the field of constraint programming. In this paper, we first recall an algorithm based on Farkas' Lemma for discovering linear ranking functions proving termination of a certain class of loops. Then we propose an extension of this method for showing the existence of *eventual linear ranking functions*, i.e., linear functions that become ranking functions after a finite unrolling of the loop. We show correctness and completeness of this polynomial algorithm.

1 Introduction et préliminaires

La terminaison des programmes est un sujet actif de recherche en informatique. Ces dernières années ont vu l'apparition d'analyseurs de terminaison performants pour des langages comme C [10] et Java [1, 13, 16] où l'emploi des techniques et outils de la programmation par contraintes est omniprésent.

Au delà des particularités des langages de programmation visés et après plusieurs abstractions (voir par exemple [16]), l'analyse de terminaison se ramène à la preuve de terminaison de boucles, dont les boucles dites *SLC* sont peut-être la catégorie la plus simple. Précisons ce type de boucles.

Une boucle SLC (pour *single-path linear constraint* [3]) sur n variables $x_1, \ldots, x_n$ est une boucle de la forme :

$$\textbf{tant que } (B\ \mathbf{x} \leq \mathbf{b}) \quad \textbf{faire } A\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} \leq \mathbf{c}$$

où $\mathbf{x} = (x_1, \ldots, x_n)^T$ et $\mathbf{x}' = (x'_1, \ldots, x'_n)^T$ sont des vecteurs colonnes de variables, B est une matrice de $\mathbb{Z}^{p\times n}$, $\mathbf{b} \in \mathbb{Z}^p$, $A \in \mathbb{Z}^{q\times 2n}$ et $\mathbf{c} \in \mathbb{Z}^q$.

Lorsque les variables sont à valeurs dans $\mathbb{Z}$ (resp. $\mathbb{Q}$), on parle de boucle *entière* (resp. *rationnelle*). Opérationnellement, de telles boucles modélisent le calcul suivant : on part d'un point $\mathbf{x}$ quelconque ; si le test $B\ \mathbf{x} \leq \mathbf{b}$ est faux, on sort de la boucle ; si le test est vrai, on choisit un nouveau point $\mathbf{x}'$ satisfaisant le corps de la boucle et on réitère en remplaçant les valeurs de $\mathbf{x}$ par celles de $\mathbf{x}'$.

Une boucle SLC peut être notée, et c'est ce que nous ferons par la suite, sous la forme d'une clause de programmation logique avec contraintes :

$$p(\mathbf{x}) \leftarrow B\mathbf{x} \leq \mathbf{b},\ A\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} \leq \mathbf{c},\ p(\mathbf{x}').$$

La terminaison des boucles en général peut toujours être établie en exhibant une *fonction de rang* ρ, c.-à-d. une fonction de $\mathbb{Z}^n$ ou $\mathbb{Q}^n$ suivant le cas vers un ensemble bien-fondé telle que à chaque passage dans la boucle la valeur $\rho(\mathbf{x})$ décroît strictement. Comme l'ensemble d'arrivée de ρ est bien-fondé, le calcul termine. À notre connaissance, la décidabilité de la terminaison universelle des boucles SLC (c.-à-d. quelque soit le point de départ du calcul et quelque soit le choix des points suivants) est une question ouverte dans le cas général. Certains cas particuliers ont été montrés décidables [6, 9, 17]. Diverses généralisations des boucles SLC sont indécidables [5].

Une approche possible pour l'analyse de la terminaison des boucles SLC consiste à restreindre la classe des fonctions de rang et le domaine des variables. Dans la section qui suit, nous présentons les résultats connus pour les fonctions de rang linéaires dans le cas $\mathbb{Q}$. Puis nous étudions les fonctions de rang linéaires à terme : ce sont les fonctions linéaires qui, après un certain nombre de passages dans la boucle, deviennent des fonctions de rang. Nous proposons un algorithme polynomial correct et complet pour cette classe de fonctions de rang, qui constitue la contribution de cet article à l'état de l'art. Enfin, nous concluons.

2 Les fonctions de rang linéaires

Précisons tout d'abord ce que nous entendons par fonction de rang linéaire.

Définition 2.1 *Soit C la boucle SLC rationnelle, exprimée sous forme clausale, $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'),\ p(\mathbf{x}')$, où p est une relation n-aire. Une* fonction de rang linéaire *ρ pour C est une application linéaire de $\mathbb{Q}^n$ vers $\mathbb{Q}$ vérifiant :*

$$\forall \mathbf{x}, \mathbf{x}' : c(\mathbf{x}, \mathbf{x}') \implies \begin{cases} \rho(\mathbf{x}) \geq 1 + \rho(\mathbf{x}'), \\ \rho(\mathbf{x}) \geq 0. \end{cases}$$

Autrement dit, ρ reste toujours positive et décroît d'au moins une unité [1] à chaque passage dans la boucle. Remarquons que si $c(\mathbf{x}, \mathbf{x}')$ est insatisfiable, la boucle termine immédiatement et toute fonction linéaire est une fonction de rang. Dans ce qui suit, nous supposerons $c(\mathbf{x}, \mathbf{x}')$ satisfiable.

Nous pouvons généraliser en considérant les fonctions de rang affines, mais cette classe de fonctions de rang peut toujours se ramener au cas linéaire. En effet, une fonction de rang affine pour la clause $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'),\ p(\mathbf{x}')$ est une fonction de rang linéaire pour $p(\mathbf{x}, y) \leftarrow c(\mathbf{x}, \mathbf{x}'), y = 1, y' = 1,\ p(\mathbf{x}', y')$ où y et y' sont des nouvelles variables distinctes de $\mathbf{x}$ et $\mathbf{x}'$. En conséquence, si on dispose d'une méthode correcte et complète pour les fonctions de rang linéaires, il en est de même pour les fonctions de rang affines. Nous nous focalisons donc sur les fonctions de rang linéaires pour les boucles SLC rationnelles et, après avoir précisé une formulation du lemme de Farkas, nous considérons la question de la vérification et de la détection.

1. Tout rationnel strictement positif fixé une fois pour toutes convient.

2.1 Lemme de Farkas

Dans les rationnels, une inéquation linéaire I est une conséquence logique d'une conjonction S finie satisfiable d'inéquations linéaires quand I est une combinaison linéaire positive des inéquations de S. Formellement, posons S :

$$\begin{cases} a_{1,1}x_1 & + \cdots & + \ a_{1,n}x_n & + \ b_1 & \geq 0 \\ \cdots & + \cdots & + \quad \cdots & + \cdots & \geq 0 \\ a_{m,1}x_1 & + \cdots & + \ a_{m,n}x_n & + \ b_m & \geq 0. \end{cases}$$

Nous supposons que S admet au moins une solution. Le lemme de Farkas établit l'équivalence entre :

$$\forall x_1, \ldots, x_n : S \implies (c_1x_1 + \cdots + c_nx_n + d \geq 0)$$

et

$$\exists \lambda_1 \geq 0, \ldots, \lambda_m \geq 0 \ . \begin{cases} c_1 = \sum_{i=1}^{m} \lambda_i a_{i,1} \\ \quad \vdots \\ c_n = \sum_{i=1}^{m} \lambda_i a_{i,n} \\ d \geq \sum_{i=1}^{m} \lambda_i b_i. \end{cases}$$

2.2 Vérification

Si on se donne une boucle SLC et une fonction ρ linéaire, on peut aisément vérifier si ρ est une fonction de rang en s'assurant du caractère insatisfiable des formules $c(\mathbf{x}, \mathbf{x}'), \rho(\mathbf{x}) < 1 + \rho(\mathbf{x}')$ et $c(\mathbf{x}, \mathbf{x}'), \rho(\mathbf{x}) < 0$. Cette vérification peut être déléguée à un solveur complet comme CLP($\mathbb{Q}$) [12]. En théorie, ce problème est polynomial.

Exemple 2.2 *Pour la boucle C :*

$$p(x, y) \leftarrow x \geq 0, y' \leq y - 1, x' \leq x + y, y \leq -1, p(x', y')$$

la fonction $\rho(x, y) = x$ est une fonction de rang linéaire, comme le montre la session Prolog ci-dessous.

```
?- use_module(library(clpq)).
% library(clpq) compiled
true.
?- {X >= 0, Y1 =< Y - 1, X1 =< X + Y, Y =< -1,
    X < 1 + X1}.
false.
?- {X >= 0, Y1 =< Y - 1, X1 =< X + Y, Y =< -1,
    X < 0}.
false.
```

2.3 Détection

Cette fois, on se donne une boucle SLC et on se demande s'il existe une fonction de rang linéaire ρ. Ce problème a été largement étudié [2, 14, 15] : il est décidable et de complexité polynomiale.

Sur l'exemple 2.2, nous explicitons un des algorithmes connus [14]. En posant $\rho(x, y) = ax + by$, exprimons formellement la question : existe-t-il une fonction de rang linéaire pour la boucle C ?

$$\exists a, b \,.\, \forall x, y, x', y' : \\ \left.\begin{array}{c} x \geq 0 \\ y' \leq y-1 \\ x' \leq x+y \\ y \leq -1 \end{array}\right\} \implies \begin{cases} ax+by \geq 1+ax'+by' \\ ax+by \geq 0. \end{cases} \tag{1}$$

Notons que cette définition du problème est *a priori* exécutable par élimination des quantificateurs sur un système de calcul formel comme Reduce [11] :

```
1: load_package redlog;
2: rlset r;
3: F:=ex({a,b},all({x,y,x1,y1},
   (x>=0 and y1<=y-1 and x1<=x+y and y<= -1)
   impl
   (a*x+b*y>=1+a*x1+b*y1 and a*x+b*y>=0)));
4: rlqe F;
```

La commande 1 charge le module d'élimination des quantificateurs. La commande 2 définit $\mathbb{R}$ comme domaine de discours. La commande 3 initialise la formule F. La commande 4 lance l'élimination des quantificateurs de la formule F et retourne une formule équivalente, ici `true`. La formule F est donc vraie : il existe au moins une fonction de rang linéaire. Déterminons les coefficients de la fonction ρ :

```
5: G:=all({x,y,x1,y1},
   (x>=0 and y1<=y-1 and x1<=x+y and y<= -1)
   impl
   (a*x+b*y>=1+a*x1+b*y1 and a*x+b*y>=0));
6: rlqe G;
```

On obtient :

$$\begin{aligned} a^2 - ab \geq 0 \wedge a - b \neq 0 \wedge a > 0 \wedge b = 0 \\ \wedge (a^2b - ab^2 \leq 0 \vee a^2 - ab = 0 \\ \vee a^2 - 2ab - a + b^2 + b \geq 0) \\ \wedge (a^2 - ab = 0 \vee a^2 - 2ab - a + b^2 b \geq 0). \end{aligned}$$

Donc tous les a et b vérifiant la formule ci-dessus conviennent. On vérifie que $a = 1, b = 0$ est bien une solution. Néanmoins, *a fortiori*, la complexité des algorithmes mis en jeu ne permettra pas systématiquement l'obtention d'une réponse avec des ressources raisonnables.

En revanche, en considérant a et b comme des *paramètres* du problème, nous pouvons appliquer le lemme de Farkas. Pour la condition de décroissance de la fonction de rang :

$$\forall x, y, x', y' : \\ \left.\begin{array}{c} x \geq 0 \\ y' \leq y-1 \\ x' \leq x+y \\ y \leq -1 \end{array}\right\} \implies ax+by \geq 1+ax'+by'$$

en faisant apparaître les coefficients λ_i sur la colonne de gauche de façon à faciliter son emploi, le lemme de Farkas s'écrit :

$$\begin{array}{lrrrrrl} \lambda_1 : & 1x & +0y & +0x' & +0y' & +0 & \geq 0 \\ \lambda_2 : & 1x & +1y & -1x' & +0y' & +0 & \geq 0 \\ \lambda_3 : & 0x & +1y & +0x' & -1y' & -1 & \geq 0 \\ \lambda_4 : & 0x & -1y & +0x' & +0y' & -1 & \geq 0 \\ \implies & & & & & & \\ & ax & +by & -ax' & -by' & -1 & \geq 0. \end{array}$$

La condition de décroissance est donc équivalente à l'existence de quatre rationnels positifs ou nuls λ_1, ..., λ_4 tels que :

$$\begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - \lambda_4. \end{cases} \tag{2}$$

Pour la condition de positivité de la fonction de rang :

$$\forall x, y, x', y' : \left\{\begin{array}{c} x \geq 0 \\ y' \leq y-1 \\ x' \leq x+y \\ y \leq -1 \end{array}\right\} \implies ax+by \geq 0$$

le lemme de Farkas s'écrit :

$$\begin{array}{lrrrrrl} \lambda'_1 : & 1x & +0y & +0x' & +0y' & +0 & \geq 0 \\ \lambda'_2 : & 1x & +1y & -1x' & +0y' & +0 & \geq 0 \\ \lambda'_3 : & 0x & +1y & +0x' & -1y' & -1 & \geq 0 \\ \lambda'_4 : & 0x & -1y & +0x' & +0y' & -1 & \geq 0 \\ \implies & & & & & & \\ & ax & +by & +0x' & +0y' & +0 & \geq 0. \end{array}$$

La condition de positivité est cette fois équivalente à l'existence de quatre autres rationnels positifs ou nuls

λ'_1, …, λ'_4 tels que :

$$\begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - \lambda'_4. \end{cases} \tag{3}$$

En résumé, via le lemme de Farkas, la formule (1) est équivalente à la conjonction des formules (2) et (3) :

$$\exists a, b \;.\; \exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0, \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0 \;.$$
$$\begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - \lambda_4 \\ a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - \lambda'_4. \end{cases}$$

En théorie, le problème de l'existence d'une fonction de rang linéaire est polynomial, ainsi que le calcul d'une fonction témoin, *un certificat* de terminaison, puisque le calcul d'une solution quelconque donne des valeurs à a et b qui répondent au problème. Si on s'intéresse à l'ensemble des fonctions de rang linéaires, c.-à-d. aux conditions sur a et b, il suffit d'éliminer les λ_i et λ'_i du système précédent par exemple en utilisant l'algorithme de Fourier-Motzkin. En voici une illustration. En compilant ce programme :

```
fm(A, B) :-
   {L1 >= 0, L2 >= 0, L3 >= 0, L4 >= 0,
    LP1 >= 0, LP2 >= 0, LP3 >= 0, LP4 >= 0,
    A = L1 + L2,
    B = L2 + L3 - L4,
    A = L2,
    B = L3,
    -1 >= -L3 - L4,
    A = LP1 + LP2,
    B = LP2 + LP3 - LP4,
    0 = LP2,
    0 = LP3,
    0 >= -LP3 - LP4}.
```

puis en interrogeant Prolog, on obtient la condition :

```
| ?- fm(A, B).
B = 0, {A >= 1}.
| ?-
```

qu'on peut montrer équivalente à la condition générée par Reduce, une fois simplifiée.

3 Les fonctions de rang linéaires à terme

Dans la section précédente, nous avons exposé une méthode qui permet de décider de l'existence d'une fonction de rang linéaire pour une boucle SLC rationnelle. Il est bien sûr possible que la méthode échoue.

Exemple 3.1 *La boucle :*

$$p(x, y) \leftarrow x \geq 0, y' \leq y - 1, x' \leq x + y, p(x', y')$$

n'admet pas de fonction de rang linéaire.

Peut-on en conclure que cette boucle ne termine pas ? Non, car il existe peut-être une fonction de rang non-linéaire. Dans cette section, nous allons étendre la méthode précédente pour la détection de fonctions de rang linéaires à terme, c.-à-d. des fonctions linéaires qui deviennent des fonctions de rang *après un nombre fini de passages dans le corps de la boucle.* Pour modéliser ce point, nous allons supposer que la boucle SLC considérée est donnée avec une fonction linéaire $f(x, y)$ dont la valeur croît strictement à chaque passage dans le corps de la boucle.

Exemple 3.2 *La fonction $f(x, y) = -y$ croît strictement pour la boucle de l'exemple 3.1 puisque y', la nouvelle valeur de y, est nécessairement inférieure d'au moins une unité à celle de y.*

Définition 3.3 *Soient C la boucle SLC rationnelle, sous forme clausale : $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}')$, $p(\mathbf{x}')$, où p est une relation n-aire, et $f(\mathbf{x})$ une fonction linéaire vérifiant $\forall \mathbf{x}, \mathbf{x}' : c(\mathbf{x}, \mathbf{x}') \implies f(\mathbf{x}') \geq 1 + f(\mathbf{x})$. Une* fonction de rang linéaire à terme *ρ pour (C, f) est une application linéaire de $\mathbb{Q}^n$ vers $\mathbb{Q}$ vérifiant :*

$$\exists k \;.\; \forall \mathbf{x}, \mathbf{x}' : \left\{ \begin{array}{l} c(\mathbf{x}, \mathbf{x}') \\ f(\mathbf{x}) \geq k \end{array} \right\} \implies \begin{cases} \rho(\mathbf{x}) \geq 1 + \rho(\mathbf{x}') \\ \rho(\mathbf{x}) \geq 0. \end{cases}$$

Relativement à la définition 2.1, le seuil k est quantifié existentiellement et on impose $f(\mathbf{x}) \geq k$ dans la partie gauche de l'implication.

Notons que si nous déterminons un tel rationnel k, alors tout $k' \geq k$ vérifie également la définition 3.3. D'autre part, comme par hypothèse la valeur de f croît d'une unité à chaque passage dans la boucle, si f est bornée supérieurement pour la contrainte c, la boucle termine. Sinon, après un nombre fini de passages dans la boucle, la valeur de f dépasse le seuil k, ρ devient alors une fonction de rang linéaire au sens de la section 2 et la boucle termine. Nous nous intéressons à présent au problème de la détection puis de la vérification des fonctions de rang linéaires à terme.

3.1 Détection

Considérons la boucle de l'exemple 3.1 associée à la fonction croissante de l'exemple 3.2. En posant $\rho(x, y) = ax + by$, ρ est une fonction de rang linéaire à terme quand :

$$\exists a, b, k \;.\; \forall x, y, x', y' : \quad \left.\begin{array}{r} x \geq 0 \\ y' \leq y - 1 \\ x' \leq x + y \\ -y \geq k \end{array}\right\} \implies \begin{cases} ax + by \geq 1 + ax' + by' \\ ax + by \geq 0. \end{cases}$$

Cette définition du problème, que nous noterons $\exists a, b, k \;.\; \phi(a, b, k)$, est aussi exécutable par élimination des quantificateurs, donc le problème est décidable. En considérant a, b et k comme des paramètres, appliquons le lemme de Farkas :

$$\begin{array}{lrrrrrl}
\lambda_1 : & 1x & +0y & +0x' & +0y' & +0 & \geq 0 \\
\lambda_2 : & 1x & +1y & -1x' & +0y' & +0 & \geq 0 \\
\lambda_3 : & 0x & +1y & +0x' & -1y' & -1 & \geq 0 \\
\lambda_4 : & 0x & -1y & +0x' & +0y' & -k & \geq 0 \\
\implies & & & & & & \\
& ax & +by & -ax' & -by' & -1 & \geq 0 \\
& ax & +by & & & & \geq 0.
\end{array}$$

On en déduit que $\phi(a, b, k)$ est équivalente à la conjonction $DEC(a, b, k)$:

$$\exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0 \;.\; \begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - k\lambda_4 \end{cases}$$

assurant la décroissance de la fonction de rang et $POS(a, b, k)$:

$$\exists \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0 \;.\; \begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - k\lambda'_4 \end{cases}$$

assurant la positivité de la fonction de rang.

Examinons $DEC(a, b, k)$. Nous nous apercevons que le produit $k\lambda_4$ introduit une non-linéarité que nous pouvons éliminer en remarquant que, comme $\lambda_4 \geq 0$, soit $\lambda_4 = 0$ (et donc $k\lambda_4 = 0$) soit $\lambda_4 > 0$. Pour ce dernier cas nous introduisons la variable $P = k\lambda_4$. On a alors la propriété :

Lemme 3.4 *La formule $\exists k\, DEC(a, b, k)$ est équivalente à la disjonction $DEC_1(a, b) \vee DEC_2(a, b)$.*

Dans notre cas, $DEC_1(a, b)$ équivaut à :

$$\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0 \;.\; \begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 \end{cases}$$

et $DEC_2(a, b)$ à :

$$\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, \lambda_4 > 0, P \;.\; \begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - P. \end{cases}$$

Preuve ($\Rightarrow$) Soient k un rationnel et quatre rationnels positifs ou nuls λ_i pour $1 \leq i \leq 4$ vérifiant la définition de $DEC(a, b, k)$. Si $\lambda_4 = 0$, alors $DEC(a, b, k)$ se simplifie en $DEC_1(a, b)$ qui est vraie. Si $\lambda_4 > 0$, en posant $P = k\lambda_4$, nous constatons que $DEC_2(a, b)$ est vraie.

($\Leftarrow$) Supposons $DEC_1(a, b)$ vraie. Alors en prenant $\lambda_4 = 0$ et k rationnel quelconque, 0 par exemple, il existe k tel que $DEC(a, b, k)$ est vraie. Supposons $DEC_2(a, b)$ vraie. Posons $k = P/\lambda_4$ (toujours possible puisque $\lambda_4 > 0$) : nous constatons qu'il existe k tel que $DEC(a, b, k)$ est vraie.

Concernant la condition de positivité, nous avons également :

Lemme 3.5 *La formule $\exists k \;.\; POS(a, b, k)$ est équivalente à la disjonction $POS_1(a, b) \vee POS_2(a, b)$.*

Dans notre cas, $POS_1(a, b)$ équivaut à :

$$\exists \lambda'_1 \geq 0, \lambda'_2 \geq 0, \lambda'_3 \geq 0 \;.\; \begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 \end{cases}$$

et $POS_2(a, b)$ à :

$$\exists \lambda'_1 \geq 0, \lambda'_2 \geq 0, \lambda'_3 \geq 0, \lambda'_4 > 0, P' \;.\; \begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - P'. \end{cases}$$

Nous pouvons à présent combiner les deux résultats précédents :

Proposition 3.6 *La formule $\exists k \,.\, \phi(a,b,k)$ équivaut à $[DEC_1(a,b) \vee DEC_2(a,b)] \wedge [POS_1(a,b) \vee POS_2(a,b)]$.*

Preuve D'après les deux lemmes précédents, il reste à justifier l'équivalence entre les formules $\exists k \,.\, \phi(a,b,k)$ et $\exists k \,.\, DEC(a,b,k) \wedge \exists k \,.\, POS(a,b,k)$.

($\Rightarrow$) Soit k_0 un rationnel vérifiant $\phi(a,b,k_0)$. Comme $\phi(a,b,k) \iff DEC(a,b,k) \wedge POS(a,b,k)$, on a $DEC(a,b,k_0)$ et $POS(a,b,k_0)$.

($\Leftarrow$) Supposons l'existence de k_d tel que $DEC(a,b,k_d)$ et k_p tel que $POS(a,b,k_p)$. Alors $k_0 = \max(k_d,k_p)$ valide $DEC(a,b,k_0) \wedge POS(a,b,k_0)$ et montre $\exists k \,.\, \phi(a,b,k)$.

Pour revenir à notre problème initial, l'existence d'une fonction de rang linéaire à terme se résume donc à la satisfiabilité d'au moins un des quatre systèmes linéaires :

$$\begin{array}{l} DEC_1(a,b) \wedge POS_1(a,b) \\ DEC_1(a,b) \wedge POS_2(a,b) \\ DEC_2(a,b) \wedge POS_1(a,b) \\ DEC_2(a,b) \wedge POS_2(a,b) \end{array}$$

ce que nous pouvons décider en temps polynomial. Pour notre exemple courant, il s'agit de $DEC_2(a,b) \wedge POS_1(a,b)$, ce que confirme la requête Prolog suivante :

```
?- dec2pos1.
true.
?-
```

après compilation du programme :

```
dec2pos1 :-
   {L1 >= 0, L2 >= 0, L3 >= 0, L4 > 0,
       A = L1 + L2,
       B = L2 + L3 - L4,
       A = L2,
       B = L3,
       1 =< L3 + P,
    LP1 >= 0, LP2 >= 0, LP3 >= 0,
       A = LP1 + LP2,
       B = LP2 + LP3,
       0 = LP2,
       0 = LP3,
       0 =< LP3}.
```

Nous sommes maintenant en mesure de définir un algorithme de détection d'existence d'une fonction de rang linéaire à terme (cf. l'algorithme 1).

Théorème 3.7 *Pour toute boucle SLC C et toute fonction linéaire strictement croissante associée à C, l'algorithme 1 décide l'existence d'une fonction de rang linéaire à terme en temps polynomial.*

Algorithme 1 Existence d'une fonction de rang linéaire à terme

Entrée : Une boucle SLC $p(\mathbf{x}) \leftarrow c(\mathbf{x},\mathbf{x}'), p(\mathbf{x}')$ et $f(\mathbf{x})$ une fonction linéaire vérifiant la condition $\forall \mathbf{x},\mathbf{x}' : c(\mathbf{x},\mathbf{x}') \implies f(\mathbf{x}') \geq 1 + f(\mathbf{x})$.
Sortie : Décide l'existence d'une fonction de rang à terme $\rho(\mathbf{x}) = \mathbf{ax} = \sum a_i x_i$ à partir du seuil k.

$DEC(\mathbf{a},k) \leftarrow$ Farkas pour la décroissance de ρ
$DEC_1(\mathbf{a}), DEC_2(\mathbf{a}) \leftarrow$ linéarisat. de $DEC(\mathbf{a},k)$
$POS(\mathbf{a},k) \leftarrow$ Farkas pour la positivité de ρ
$POS_1(\mathbf{a}), POS_2(\mathbf{a}) \leftarrow$ linéarisation de $POS(\mathbf{a},k)$
si $\bigvee_{1 \leq i,j \leq 2} DEC_i(\mathbf{a}) \wedge POS_j(\mathbf{a})$ est satisfiable **alors**
  retourner vrai
sinon
  retourner faux
fin si

Le calcul d'un certificat de terminaison, c.-à-d. d'une fonction de rang linéaire à terme ρ et du seuil k associé peut s'effectuer lors de l'exécution de l'algorithme comme suit :

- si $DEC_1(\mathbf{a}) \wedge POS_1(\mathbf{a})$ est satisfiable, on calcule une solution $\mathbf{a}$ de la conjonction et tout rationnel k convient ;
- si $DEC_1(\mathbf{a}) \wedge POS_2(\mathbf{a})$ est satisfiable, on calcule une solution $\mathbf{a}$, λ', P' de la conjonction, on prend $k = P'/\lambda'_n$;
- si $DEC_2(\mathbf{a}) \wedge POS_1(\mathbf{a})$ est satisfiable, on calcule une solution $\mathbf{a}$, λ, P de la conjonction, on prend $k = P/\lambda_n$;
- si $DEC_2(\mathbf{a}) \wedge POS_2(\mathbf{a})$ est satisfiable, on calcule une solution $\mathbf{a}$, λ, P, λ', P' de la conjonction, on prend $k = \max(P/\lambda_n, P'/\lambda'_n)$.

Exemple 3.8 *Poursuivons l'exemple 3.1. Voici la solution la plus générale de $DEC_2(a,b) \wedge POS_1(a,b)$*

```
?-  {L1 >= 0, L2>= 0, L3 >= 0, L4 > 0,
        A = L1 + L2,
        B = L2 + L3 - L4,
        A = L2,
        B = L3,
        1 =< L3 + P,
     LP1 >= 0, LP2 >= 0, LP3 >= 0,
        A = LP1 + LP2,
        B = LP2 + LP3,
        0 = LP2,
        0 = LP3,
        0 =< LP3}.
B = 0, L1 = 0, L3 = 0, LP2 = 0, LP3 = 0,
{LP1 = L4, L2 = L4, A = L4, L4 > 0, P >= 1}.
?-
```

Une solution particulière est $b = 0 = \lambda_1 = \lambda_3 = \lambda'_2 =$

λ'_3, $a = 1 = \lambda'_1 = \lambda_2 = \lambda_4$, $P = 1$. *Nous en déduisons que $\rho(x,y) = x$ est une fonction de rang linéaire à terme à partir du seuil $k = P/\lambda_4 = 1$.*

3.2 Vérification

Étant données une boucle SLC C, une fonction strictement croissante associée f et une fonction linéaire ρ, il s'agit à présent de déterminer si ρ est bien une fonction de rang à terme. La vérification peut s'effectuer en appliquant l'algorithme 1, avec les coefficients $\mathbf{a}$ instanciés comme précisé par ρ. On peut au besoin déterminer le seuil k comme indiqué précédemment.

3.3 D'autres exemples

Les exemples qui suivent complémentent ceux de la partie 3.1. Nous cherchons des fonctions de rang linéaires à terme pour ces boucles SLC. Nous nous restreignons aux fonctions strictement croissantes de la forme $f(x_1,\dots,x_n) = x_i$ et $f(x_1,\dots,x_n) = -x_i$, aisément détectables via une technique similaire à celle exposée section 2.2.

Exemple 3.9 *Étudions la boucle suivante :*

$$\begin{aligned} p(x,y,z) \leftarrow\ & y' = 10,\ y - z \geq 1,\\ & x = z,\ z - z' \leq -1,\ x' = z',\\ & p(x',y',z') \end{aligned}$$

qui n'admet pas de fonction de rang linéaire. La fonction $f(x,y,z) = z$ croît strictement pour cette boucle puisque z', la nouvelle valeur de z, est nécessairement supérieure d'une unité à celle de z.

En posant $\rho(x,y,z) = ax+by+cz$, ρ est une fonction de rang linéaire à terme quand :

$$\exists a,b,c,k\ .\ \forall x,y,z,x',y',z' :$$

$$\left.\begin{array}{r} y' \geq 10 \\ y' \leq 10 \\ y - z \geq 1 \\ x \geq z \\ x \leq z \\ z - z' \leq -1 \\ x' \geq z' \\ x' \leq z' \\ z \geq k \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} ax+by+cz \\ \qquad \geq 1 + ax' + by' + cz' \\ ax+by+cz \geq 0. \end{array}\right.$$

En considérant a, b, c et k comme des paramètres et en appliquant le lemme de Farkas ainsi que les lemmes 3.4 et 3.5, nous obtenons :

– $DEC_1(a,b,c)$:

$$\exists \lambda_1 \geq 0, \dots, \lambda_8 \geq 0\ .$$
$$\begin{cases} a = \lambda_4 - \lambda_5 \\ b = \lambda_3 \\ c = -\lambda_3 - \lambda_4 + \lambda_5 - \lambda_6 \\ -a = \lambda_7 - \lambda_8 \\ -b = \lambda_1 - \lambda_2 \\ -c = \lambda_6 - \lambda_7 + \lambda_8 \\ -1 \geq -10\lambda_1 + 10\lambda_2 - \lambda_3 - \lambda_6 \end{cases}$$

– $DEC_2(a,b,c)$:

$$\exists \lambda_1 \geq 0, \dots, \lambda_8 \geq 0, \lambda_9 > 0, P\ .$$
$$\begin{cases} a = \lambda_4 - \lambda_5 \\ b = \lambda_3 \\ c = -\lambda_3 - \lambda_4 + \lambda_5 - \lambda_6 + \lambda_9 \\ -a = \lambda_7 - \lambda_8 \\ -b = \lambda_1 - \lambda_2 \\ -c = \lambda_6 - \lambda_7 + \lambda_8 \\ -1 \geq -10\lambda_1 + 10\lambda_2 - \lambda_3 - \lambda_6 - P \end{cases}$$

– $POS_1(a,b,c)$:

$$\exists \lambda'_1 \geq 0, \dots, \lambda'_8 \geq 0\ .$$
$$\begin{cases} a = \lambda'_4 - \lambda'_5 \\ b = \lambda'_3 \\ c = -\lambda'_3 - \lambda'_4 + \lambda'_5 - \lambda'_6 \\ 0 = \lambda'_7 - \lambda'_8 \\ 0 = \lambda'_1 - \lambda'_2 \\ 0 = \lambda'_6 - \lambda'_7 + \lambda'_8 \\ 0 \geq -10\lambda'_1 + 10\lambda'_2 - \lambda'_3 - \lambda'_6 \end{cases}$$

– $POS_2(a,b,c)$:

$$\exists \lambda'_1 \geq 0, \dots, \lambda'_8 \geq 0, \lambda'_9 > 0, P'\ .$$
$$\begin{cases} a = \lambda'_4 - \lambda'_5 \\ b = \lambda'_3 \\ c = -\lambda'_3 - \lambda'_4 + \lambda'_5 - \lambda'_6 + \lambda'_9 \\ 0 = \lambda'_7 - \lambda'_8 \\ 0 = \lambda'_1 - \lambda'_2 \\ 0 = \lambda'_6 - \lambda'_7 + \lambda'_8 \\ 0 \geq -10\lambda'_1 + 10\lambda'_2 - \lambda'_3 - \lambda'_6 - P'. \end{cases}$$

La formule $DEC_2(a,b,c) \wedge POS_2(a,b,c)$ admet une solution, par exemple : $\lambda_6 = \lambda'_5 = \lambda'_6 = 0$, $a = b = 1 = \lambda_1 = \lambda_3 = \lambda_5 = \lambda_7 = \lambda_9 = \lambda'_1 = \lambda'_2 = \lambda'_3 = \lambda'_4 = \lambda'_7 = \lambda'_8 = \lambda'_9$, $\lambda_2 = \lambda_4 = \lambda_8 = 2$, $c = -1$, et $P = P' = 11$.

Nous en déduisons que $\rho(x,y,z) = x + y - z$ est bien une fonction de rang linéaire à terme à partir du seuil $k = P/\lambda_9 = P'/\lambda'_9 = 11$.

Exemple 3.10 *Étudions la boucle de l'exemple 2.2 et montrons que cette boucle admet également une fonction de rang linéaire à terme. La fonction $f(x,y) = -y$ croît strictement pour cette boucle car y', la nouvelle valeur de y, est nécessairement inférieure d'une unité à celle de y.*

En posant $\rho(x,y) = ax + by$, ρ est une fonction de rang linéaire à terme quand :

$$\exists a,b,k\ .\ \forall x,y,x',y' :\quad \left.\begin{array}{r} x \geq 0 \\ y' \leq y - 1 \\ x' \leq x + y \\ y \leq -1 \\ -y \geq k \end{array}\right\} \implies \begin{cases} ax + by \geq 1 + ax' + by' \\ ax + by \geq 0. \end{cases}$$

En considérant a ,b et k comme des paramètres et en appliquant le lemme de Farkas ainsi que les lemmes 3.4 et 3.5, nous obtenons :

– $DEC_1(a,b)$:

$$\exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0\ .\ \begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - \lambda_4 \end{cases}$$

– $DEC_2(a,b)$:

$$\exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0, \lambda_5 > 0, P\ .\ \begin{cases} a = \lambda_1 + \lambda_2 \\ b = \lambda_2 + \lambda_3 - \lambda_4 - \lambda_5 \\ -a = -\lambda_2 \\ -b = -\lambda_3 \\ -1 \geq -\lambda_3 - \lambda_4 - P \end{cases}$$

– $POS_1(a,b)$:

$$\exists \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0\ .\ \begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - \lambda'_4 \end{cases}$$

– $POS_2(a,b)$:

$$\exists \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0, \lambda'_5 > 0, P'\ .\ \begin{cases} a = \lambda'_1 + \lambda'_2 \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4 - \lambda'_5 \\ 0 = -\lambda'_2 \\ 0 = -\lambda'_3 \\ 0 \geq -\lambda'_3 - \lambda'_4 - P'. \end{cases}$$

La formule $DEC_1(a,b) \wedge POS_1(a,b)$ admet une solution, par exemple : $b = 0 = \lambda_1 = \lambda_3 = \lambda'_2 = \lambda'_3 = \lambda'_4$, $a = 1 = \lambda_2 = \lambda_4 = \lambda'_1$. Il n'y pas de condition sur k, donc $k = 0$ convient. Nous en déduisons que $\rho(x,y) = x$ est une fonction de rang linéaire à terme à partir du seuil $k = 0$.

Exemple 3.11 *Étudions la boucle suivante :*

$$p(x) \leftarrow x = -1,\ x' = -2,\ p(x').$$

La fonction $f(x) = -x$ croît strictement pour cette boucle.

En posant $\rho(x) = ax$, ρ est une fonction de rang linéaire à terme quand :

$$\exists a,k\ .\ \forall x,x' : \left\{\begin{array}{r} x \geq -1 \\ x \leq -1 \\ x' \geq -2 \\ x' \leq -2 \\ -x \geq k \end{array}\right\} \implies \begin{cases} ax \geq 1 + ax' \\ ax \geq 0. \end{cases}$$

En considérant a et k comme des paramètres et en appliquant le lemme de Farkas ainsi que les lemmes 3.4 et 3.5, nous obtenons :

– $DEC_1(a)$:

$$\exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0\ .\ \begin{cases} a = \lambda_1 - \lambda_2 \\ -a = \lambda_3 - \lambda_4 \\ -1 \geq \lambda_1 - \lambda_2 + 2\lambda_3 - 2\lambda_4 \end{cases}$$

– $DEC_2(a)$:

$$\exists \lambda_1 \geq 0, \ldots, \lambda_4 \geq 0, \lambda_5 > 0, P\ .\ \begin{cases} a = \lambda_1 - \lambda_2 - \lambda_5 \\ -a = \lambda_3 - \lambda_4 \\ -1 \geq \lambda_1 - \lambda_2 + 2\lambda_3 - 2\lambda_4 - P \end{cases}$$

– $POS_1(a)$:

$$\exists \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0\ .\ \begin{cases} a = \lambda'_1 - \lambda'_2 \\ 0 = \lambda'_3 - \lambda'_4 \\ 0 \geq \lambda'_1 - \lambda'_2 + 2\lambda'_3 - 2\lambda'_4 \end{cases}$$

– $POS_2(a)$:

$$\exists \lambda'_1 \geq 0, \ldots, \lambda'_4 \geq 0, \lambda'_5 > 0, P' \; . \begin{cases} a = \lambda'_1 - \lambda'_2 - \lambda'_5 \\ 0 = \lambda'_3 - \lambda'_4 \\ 0 \geq \lambda'_1 - \lambda'_2 + 2\lambda'_3 - 2\lambda'_4 - P'. \end{cases}$$

La formule $DEC_1(a) \wedge POS_2(a)$ admet une solution, par exemple : $a = 1 = \lambda_1 = \lambda_4 = \lambda'_5$, $\lambda_2 = 0 = \lambda_3 = \lambda'_2 = \lambda'_3 = \lambda'_4$, $\lambda'_1 = 2 = P'$. Nous en déduisons que $\rho(x) = x$ est une fonction de rang linéaire à terme à partir du seuil $k = P'/\lambda'_5 = 2$.

4 Conclusion

Dans cet article, pour les boucles SLC rationnelles, nous avons proposé une définition des fonctions de rang linéaires à terme assortie d'un algorithme polynomial correct et complet pour la détection de telles fonctions de rang.

Cet algorithme est implanté dans l'analyseur de terminaison BinTerm [16]. Il admet une formulation élégante en programmation logique avec contraintes. Il nous reste à approfondir le choix de la "bonne" fonction croissante. Pour le moment, nous nous restreignons aux fonctions candidates de la forme $f(x_1, \ldots, x_n) = x_i$ ou $-x_i$.

Le concept d'*eventual termination*, similaire à la notion de fonction de rang à terme, apparaît dans [7, 8]. La classe de boucles qui y est abordée est plus vaste, mais comme les auteurs s'appuient sur des techniques de différences finies, les approches sont incomplètes.

Concernant les fonctions de rang linéaires des boucles SLC entières, la *vérification*, c.-à-d. déterminer dans $\mathbb{Z}$ la satisfiabilité des formules $c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 1 + \rho(\mathbf{x}')$ et $c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 0$, est à présent un problème de programmation linéaire en nombres entiers, de complexité NP. Pour la *détection*, comme le lemme de Farkas n'est plus vrai dans $\mathbb{Z}$, la méthode exposée à la section 2 n'est pas directement transposable. La question de l'existence d'une fonction de rang linéaire dans $\mathbb{Z}$ vient d'être résolue très récemment par [4]. Le problème appartient à la classe coNP et les auteurs présentent un algorithme de complexité exponentielle qui génère les fonctions de rang linéaires.

L'étude des fonctions de rang linéaires à terme pour les boucles SLC entières est une extension naturelle à notre travail à envisager.

Références

[1] E. Albert, P. Arenas, S. Genaim, M. Gómez-Zamalloa, G. Puebla, D. V. Ramírez, G. Román, and D. Zanardini. Termination and cost analysis with COSTA and its user interfaces. *Electronic Notes in Theoretical Computer Science*, 258(1) :109–121, 2009.

[2] R. Bagnara, F. Mesnard, A. Pescetti, and E. Zaffanella. A new look at the automatic synthesis of linear ranking functions. *Information and Computation*, 215 :47–67, 2012.

[3] A. M. Ben-Amram and S. Genaim. On the linear ranking problem for integer linear-constraint loops. Technical Report arXiv :1208.4041v2 [cs.PL], 2013. Available from `http://arxiv.org/`.

[4] A. M. Ben-Amram and S. Genaim. On the linear ranking problem for integer linear-constraint loops. In R. Giacobazzi and R. Cousot, editors, *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '13)*, pages 51–62, Rome, Italy, 2013. Association for Computing Machinery.

[5] A. M. Ben-Amram, S. Genaim, and A. N. Masud. On the termination of integer loops. *ACM Transactions on Programming Languages and Systems*, 34(4) :16 :1–16 :24, 2012.

[6] M. Bozga, R. Iosif, and F. Konecný. Deciding conditional termination. In C. Flanagan and B. König, editors, *Tools and Algorithms for the Construction and Analysis of Systems : Proceedings of the 18th International Conference (TACAS 2012)*, volume 7214 of *Lecture Notes in Computer Science*, pages 252–266, Tallinn, Estonia, 2012. Springer.

[7] A. R. Bradley, Z. Manna, and H. B. Sipma. The polyranking principle. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming : Proceedings of the 32nd International Colloquium (ICALP 2005)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1349–1361, Lisbon, Portugal, 2005. Springer.

[8] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination of polynomial programs. In R. Cousot, editor, *Verification, Model Checking and Abstract Interpretation : Proceedings of the 6th International Conference (VMCAI 2005)*, volume 3385 of *Lecture Notes in Computer Science*, pages 113–129, Paris, France, 2005. Springer-Verlag, Berlin.

[9] M. Braverman. Termination of integer linear programs. In T. Ball and R. B. Jones, editors, *Computer Aided Verification : Proceedings of the 18th International Conference (CAV 2006)*, vol-

ume 4144 of *Lecture Notes in Computer Science*, pages 372–385, Seattle, WA, USA, 2006. Springer.

[10] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In M. I. Schwartzbach and T. Ball, editors, *Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation*, pages 415–426, Ottawa, Ontario, Canada, 2006. Association for Computing Machinery.

[11] A. C. Hearn. REDUCE : the first forty years. In A. Dolzmann, A. Seidl, and T. Sturm, editors, *Algorithmic Algebra and Logic : Proceedings of the A3L 2005 Conference in Honor of the 60th Birthday of Volker Weispfenning*, pages 19–24, Passau, Germany, 2005.

[12] C. Holzbaur. *OFAI clp(Q,R)*. Austrian Research Institute for Artificial Intelligence, Vienna, 1.3.3 edition, 1995. Published as TR-95-09.

[13] C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl. Automated termination analysis of Java bytecode by term rewriting. In C. Lynch, editor, *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 259–276, Edinburgh, Scotland, UK, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[14] A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In B. Steffen and G. Levi, editors, *Verification, Model Checking and Abstract Interpretation : Proceedings of the 5th International Conference (VMCAI 2004)*, volume 2937 of *Lecture Notes in Computer Science*, pages 239–251, Venice, Italy, 2004. Springer.

[15] K. Sohn and A. Van Gelder. Termination detection in logic programs using argument sizes (extended abstract). In D. J. Rosenkrantz, editor, *Proceedings of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 216–226, Denver, CO, USA, 1991. Association for Computing Machinery.

[16] F. Spoto, F. Mesnard, and É. Payet. A termination analyzer for Java bytecode based on path-length. *ACM Transactions on Programming Languages and Systems*, 32(3), 2010.

[17] A. Tiwari. Termination of linear programs. In R. Alur and D. Peled, editors, *Computer Aided Verification : Proceedings of the 16th International Conference (CAV 2004)*, volume 3114 of *Lecture Notes in Computer Science*, pages 70–82, Boston, MA, USA, 2004. Springer.