Eventual Linear Ranking Functions

Roberto Bagnara
BUGSENG (http://bugseng.com)
Dipartimento di Matematica e Informatica
Università di Parma, Italy
bagnara@cs.unipr.it

Fred Mesnard
LIM
Université de la Réunion, France
frederic.mesnard@univ-reunion.fr

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 declarative 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 algorithm.

Keywords

termination analysis, ranking function, eventual linear ranking function.

1. INTRODUCTION

Program termination is a hot research topic in program analysis. The last few years have witnessed the development of termination analyzers for mainstream programming languages such as C [16] and Java [1, 21, 24] with remarkable precision and performance. These systems are largely based on techniques and tools coming from the field of declarative constraint programming.

Beyond the specificities of the targeted programming languages and after several abstractions (see, e.g., [24]), termination analysis of entire programs boils down to termination analysis of individual loops. Various categories of loops have been identified: for the purposes of this paper we focus on *single-path linear constraint* (SLC) loops [9].

We first recall some notions and notations. For $n \in \mathbb{N}$, we denote by $\mathbb{Q}^n$ the n-dimensional vector space on the field of rational numbers $\mathbb{Q}$.

PPDP '13, September 16 – 18 2013, Madrid, Spain

http://dx.doi.org/10.1145/2505879.2505884 .

A vector $\mathbf{v} \in \mathbb{Q}^n$ can be also interpreted as a matrix in $\mathbb{Q}^{n\times 1}$ and manipulated accordingly using the usual definitions for addition, multiplication (both by a scalar and by another matrix), and transposition, denoted by $\mathbf{v}^{\mathrm{T}}$. For each $i = 1, \ldots\ n$, v_i denotes the i-th component of the (column) vector $\mathbf{v} = (v_1, \ldots, v_n)^{\mathrm{T}} \in \mathbb{Q}^n$. The *scalar product* of $\mathbf{v}, \mathbf{w} \in \mathbb{Q}^n$, denoted $\langle \mathbf{v}, \mathbf{w} \rangle$, is the rational number

$$\mathbf{v}^{\mathrm{T}}\mathbf{w} = \sum_{i=1}^{n} v_i w_i.$$

Now, an SLC loop over n variables $x_1, \ldots, x_n$ has the form

$$\textbf{while } (B\ \mathbf{x} \leq \mathbf{b}) \quad \textbf{do } A \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq \mathbf{c}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{x}' = (x'_1, \ldots, x'_n)^T$ are column vectors of variables, $B \in \mathbb{Z}^{p\times n}$ is an integer matrix, $\mathbf{b} \in \mathbb{Z}^p$, $A \in \mathbb{Z}^{q\times 2n}$ and $\mathbf{c} \in \mathbb{Z}^q$. Such a loop can be conveniently written as a constraint logic programming rule:

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

When variables take their values in $\mathbb{Z}$ (resp., $\mathbb{Q}$), we call such loops *integer* (resp., *rational*) loops. They model a computation that starts from a point $\mathbf{x}$; if $B\ \mathbf{x} \leq \mathbf{b}$ is false, the loop terminates; otherwise, a new point $\mathbf{x}'$ is chosen that satisfies

$$A \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq \mathbf{c}$$

and iteration continues replacing the values of $\mathbf{x}$ by the corresponding ones of $\mathbf{x}'$.

Loop termination can always be ensured by a *ranking function* ρ, a function from $\mathbb{Z}^n$ or $\mathbb{Q}^n$ to a well-founded set. As the range of ρ is well-founded, the computation terminates. To the best of our knowledge, decidability of universal termination of SLC loops (i.e., from any starting point and for any choice of the next point at each iteration) is an open question. Some sub-classes have been shown to be decidable [12, 15, 25]. For instance, Braverman proves that termination of loops where the body is a *deterministic* assignment $\mathbf{x}' := A\mathbf{x}$ is decidable when the variables range over $\mathbb{Q}$. The problem is open for the non-deterministic case, as stated in his paper. On the other hand, various generalizations have been shown to be undecidable [11].

A way to investigate loop termination is to restrict the class of the considered ranking functions. In the following section, we recall a well-known technique for computing linear ranking functions for rational SLC loops.

In Section 3 we present the main contribution of the paper, namely the definition of *eventual linear ranking functions*: these are linear functions that become ranking functions after a finite unrolling of the loop.[1] We shall see that the number of unrolling is not pre-defined, but depends on the data processed by the loop. Section 3 presents complete decision procedures for the existence of eventual linear ranking functions of SLC loops. The presentation is gradual and illustrates the algorithms by means of constraint logic programming (CLP) technology and dialogues with real CLP tools. Section 4 discusses related work and a preliminary experimentation conducted on the benchmarks proposed in two very recent papers. Section 5 concludes the paper.

2. LINEAR RANKING FUNCTIONS

We first define the notion of linear (resp., affine) ranking function for an SLC loop.

Definition 2.1. *Let C be the SLC loop*

$$p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}')$$

where p is an n-ary relation symbol. A linear *(resp.,* affine*)* ranking function *ρ for C is a linear (resp., affine) map from $\mathbb{Q}^n$ to $\mathbb{Q}$ such that*

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

In words, the continuation of the iteration, i.e., $c(\mathbf{x}, \mathbf{x}')$, entails that ρ stays positive and strictly decreases by at least 1 for each iteration. We point out that if $c(\mathbf{x}, \mathbf{x}')$ is not satisfiable, the loop ends immediately and any linear function is a ranking function. In the paper, we assume that $c(\mathbf{x}, \mathbf{x}')$ is satisfiable.

Remark 2.2. Definition 2.1 *might seem too restrictive when working with rational numbers as one might prefer to replace the decrease by 1 by a decrease by ε, a fixed positive quantity. Actually, by multiplying such an ε-decrease ranking function by $1/\varepsilon$, we see that the two definitions are equivalent with respect to the existence of a ranking function.*

Remark 2.3. *Although the class of affine ranking functions subsumes the class of linear ranking functions, any decision procedure for the existence of linear ranking functions can be extended to a decision procedure for the existence of affine ranking functions. To see this, note that an affine ranking function for*

$$p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'),\ p(\mathbf{x}')$$

is a linear ranking function for

$$p(\mathbf{x}, y) \leftarrow c(\mathbf{x}, \mathbf{x}'), y = 1, y' = 1,\ p(\mathbf{x}', y'),$$

where y is distinct from the variables in $\mathbf{x}$.

In this section, we focus on linear ranking functions for SLC loops. After the presentation of a formulation of Farkas' Lemma we consider the problem of verifying linear ranking functions, and then the detection of such ranking functions.

[1] A preliminary version of this work, in French, was presented in [2].

2.1 Farkas' Lemma

A linear inequation I over rational numbers is a logical consequence of a finite satisfiable conjunction S of linear inequations when I is a linear positive combination of the inequations of S. More formally, let S be

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

and suppose that S has at least one solution. Farkas' Lemma states the equivalence of

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

and

$$\exists \lambda_1 \geq 0, \ldots, \lambda_m \geq 0\ .$$
$$\left(d \geq \sum_{i=1}^{m} \lambda_i b_i\right) \wedge \bigwedge_{j=1}^{n} \left(c_j = \sum_{i=1}^{m} \lambda_i a_{i,j}\right).$$

2.2 Verification

Given an SLC loop C and a linear function ρ, we can easily check whether ρ is a ranking function for C by testing the unsatisfiability of

$$c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 1 + \rho(\mathbf{x}')$$

or

$$c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 0.$$

This test has polynomial complexity and can be done with a complete rational solver such as , e.g., CLP($\mathbb{Q}$) [20].

Example 2.4. *For the SLC loop C:*

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

the linear function $\rho(x, y) = x$ is a ranking function, as proved by the following SICStus Prolog *session.*

```
?- 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 Detection

Given an SLC loop, we would like to know whether it admits a linear ranking function ρ. This problem, which has been studied in depth [5, 6, 22, 23], is decidable in polynomial time.

Let us consider Example 2.4 and formally ask whether there exists a ranking function of the form $\rho(x, y) = ax + by$:

$$\exists a, b\ .\ \forall x, y, x', y' : \begin{Bmatrix} x \geq 0, & x' \leq x + y, \\ y \leq -1, & y' \leq y - 1 \end{Bmatrix} \implies \begin{cases} ax + by \geq 1 + ax' + by', \\ ax + by \geq 0. \end{cases} \quad (1)$$

This formulation of the problem is executable by quantifier elimination on a symbolic computation system like Reduce [19]:

```
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;
```

Statement `1` loads the quantifier elimination module. Statement `2` defines $\mathbb{R}$ as the domain of discourse. Statement `3` initializes formula F. Statement `4` runs quantifier elimination over F and returns an equivalent formula, `true` in this case. Hence, formula F is true and there exists at least one linear ranking function. We can now determine the coefficients of function ρ as follows:

```
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;
```

We obtain

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

and all values for a and b satisfying the above formula, such as $a = 1$ and $b = 0$, are equally good. Unfortunately, the complexity of the algorithms involved will prevent us from systematically obtaining such a result within acceptable time and memory bounds.

We now recall the most famous algorithm for this problem [22].[2] Considering a and b as *parameters* of the problem, we can apply Farkas' Lemma. For the strict decrease of the ranking function we have

$$\forall x, y, x', y' : \begin{Bmatrix} x \geq 0, & x' \leq x + y, \\ y \leq -1, & y' \leq y - 1 \end{Bmatrix} \implies ax + by \geq 1 + ax' + by'. \quad (2)$$

Application of Farkas' Lemma to this problem can be depicted as follows:

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

We know that formula (2) is equivalent to the existence of four non-negative rational numbers $\lambda_1, \ldots, \lambda_4$ such that:

$$\begin{cases} a = \lambda_1 + \lambda_2, & -a = -\lambda_2, \\ b = \lambda_2 + \lambda_3 - \lambda_4, & -b = -\lambda_3, \quad -1 \geq -\lambda_3 - \lambda_4. \end{cases} \quad (3)$$

[2] See also [5, 6].

The positivity of the ranking function, that is,

$$\forall x, y, x', y' : \begin{Bmatrix} x \geq 0 & x' \leq x + y \\ y \leq -1 & y' \leq y - 1 \end{Bmatrix} \implies ax + by \geq 0 \quad (4)$$

can be written as

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

By Farkas' Lemma, formula (4) is equivalent to the existence of four other non-negative rational numbers $\lambda'_1, \ldots, \lambda'_4$ such that:

$$\begin{cases} a = \lambda'_1 + \lambda'_2, & 0 = -\lambda'_2, \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4, & 0 = -\lambda'_3, \\ & 0 \geq -\lambda'_3 - \lambda'_4. \end{cases} \quad (5)$$

Summarizing, by Farkas Lemma, formula (1) is equivalent to the conjunction of formulas (3) and (5):

$$\exists a, b \,.\, \exists \lambda_1, \ldots, \lambda_4, \lambda'_1, \ldots, \lambda'_4 \geq 0 \,. \begin{cases} a = \lambda_1 + \lambda_2, & -a = -\lambda_2, \\ b = \lambda_2 + \lambda_3 - \lambda_4, & -b = -\lambda_3, \\ a = \lambda'_1 + \lambda'_2, & 0 = -\lambda'_2, \\ b = \lambda'_2 + \lambda'_3 - \lambda'_4, & 0 = -\lambda'_3, \\ -1 \geq -\lambda_3 - \lambda_4, & 0 \geq -\lambda'_3 - \lambda'_4. \end{cases} \quad (6)$$

In theory, the problem of the existence of a linear ranking function is polynomial. Since computing one solution (that is, values for a and b) is not harder than determining its existence, a "witness" function, which would constitute a *termination certificate*, can also be computed in polynomial time.

The space of all linear ranking functions as defined in Definition 2.1, described by parameters a and b, can be obtained by elimination of λ_i and λ'_i from (6) using, e.g., the algorithm of Fourier-Motzkin. For example the SICStus Prolog program

```
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}.
```

can be queried as follows:

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

It can be shown that the computed answer is equivalent to the (significantly more involved) condition generated by Reduce.

3. EVENTUAL LINEAR RANKING FUNCTIONS

In the previous section we have illustrated a method to decide the existence of a linear ranking function for a rational SLC loop, something that implies termination of the loop. Of course, the method cannot decide termination in all cases.

EXAMPLE 3.1. *The loop*

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

does not admit a linear ranking function.

Can we conclude that such loop does not always terminate? No, because it may admit a non-linear ranking function.

In this section we will extend the previous method so as to detect *eventual linear ranking functions*, that is, linear functions that behave as ranking functions *after a finite number of executions of the loop body.* Suppose that the considered SLC loop is always given with a linear function $f(x, y)$ that increases at each iteration of the loop in the following sense:

DEFINITION 3.2. *Let C be the SLC loop*

$$p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}').$$

A function $f(\mathbf{x})$ is increasing for C *if it is linear and satisfies: $\forall \mathbf{x}, \mathbf{x}' : c(\mathbf{x}, \mathbf{x}') \implies f(\mathbf{x}') \geq 1 + f(\mathbf{x})$.*

EXAMPLE 3.3. *Since y decreases by at least 1 at each iteration, the function $f(x, y) = -y$ is increasing for the loop of* Example 3.1.

REMARK 3.4. *The generalization to affine functions is useless. Moreover, as we are merely interested in the existence of an increasing function, the value of the increase (1 or $\varepsilon > 0$) is irrelevant.*

We can now give the definition which is central to our paper.

DEFINITION 3.5. *Let C be the rational SLC loop in clausal form*

$$p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}'),$$

where p is an n-ary relation; let also $f(\mathbf{x})$ be a linear increasing function for C. An eventual linear ranking function *ρ for (C, f) is a linear map of $\mathbb{Q}^n$ to $\mathbb{Q}$ such that*

$$\exists k \,.\, \forall \mathbf{x}, \mathbf{x}' : \big(c(\mathbf{x}, \mathbf{x}') \,\wedge\, f(\mathbf{x}) \geq k\big) \implies \big(\rho(\mathbf{x}) \geq 1 + \rho(\mathbf{x}') \,\wedge\, \rho(\mathbf{x}) \geq 0\big).$$

For comparison with Definition 2.1, remark that the threshold k is existentially quantified and that $f(\mathbf{x}) \geq k$ is imposed in the implication antecedent. It should also be noted that, if such a rational k exists, then each $k' \geq k$ satisfies the condition of Definition 3.5. On the other hand, since, by hypothesis, f strictly increases at each iteration, there are two cases: either f is bounded from above by a constant, and thus the loop will terminate; or, after a finite number of iterations, f will cross the threshold k and ρ becomes a linear ranking function in the sense of Section 2 so that, again, the loop terminates.

Eventual linear ranking functions are a generalization of linear ranking functions.

PROPOSITION 3.6. *Let C be an SLC loop. If ρ is a linear ranking function for C, then there exists an increasing function f such that (C, f) has an eventual linear ranking function.*

PROOF. By hypothesis, there exists a linear ranking function $\rho(\mathbf{x})$ for C. The linear function $f(\mathbf{x}) \stackrel{\text{def}}{=} -\rho(\mathbf{x})$ is nonpositive and strictly increasing for C. Considering $k = 1$ it can be seen that the function $\rho'(\mathbf{x}) \stackrel{\text{def}}{=} 0$ is an eventual linear ranking function for (C, f). $\square$

The generalization is strict as the loop of Example 3.1 has no linear ranking function, but does have an eventual linear ranking function, as will be shown in the next section.

3.1 Detection given a Linear Increasing Function

As a first step towards full automation of the synthesis of eventual linear ranking functions, we assume that an SLC loop is given with a particular linear increasing function. Let us consider, e.g., the SLC loop of Example 3.1 and the increasing function of Example 3.3. Defining $\rho(x, y) = ax + by$, ρ is an eventual linear ranking function when

$$\exists a, b, k \,.\, \forall x, y, x', y' : \begin{Bmatrix} x \geq 0, & x' \leq x + y, \\ -y \geq k, & y' \leq y - 1 \end{Bmatrix} \implies \begin{cases} ax + by \geq 1 + ax' + by', \\ ax + by \geq 0. \end{cases}$$

This definition of the problem, that we will denote for brevity with $\exists a, b, k \,.\, \phi(a, b, k)$, is also solvable via quantifier elimination, hence the problem is decidable. Considering a, b and k as parameters, we can apply Farkas' Lemma as follows:

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

Hence, formula $\phi(a, b, k)$ is equivalent to the conjunction of formulas $\mathrm{DEC}(a, b, k)$, i.e.,

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

ensuring the decreasing of the ranking function, and the formula $\mathrm{POS}(a, b, k)$, that is,

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

ensuring the positivity of the ranking function.

Let us focus on $\mathrm{DEC}(a, b, k)$. We observe that the product $k\lambda_4$ leads to a non-linearity that we can circumvent by noting that, as $\lambda_4 \geq 0$, either $\lambda_4 = 0$ (hence $k\lambda_4 = 0$) or $\lambda_4 > 0$. In the latter case, we introduce a new variable $P = k\lambda_4$. We have the property:

Lemma 3.7. *Formula* $\exists k$. $\mathrm{DEC}(a,b,k)$ *is equivalent to the disjunction* $\mathrm{DEC}_1(a,b) \vee \mathrm{DEC}_2(a,b)$.

In our case, $\mathrm{DEC}_1(a,b)$ is equivalent to

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

and $\mathrm{DEC}_2(a,b)$ is equivalent to

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

Proof. ($\Longrightarrow$) Let k be a rational number and λ_i's for $1 \leq i \leq 4$ four non-negative rational numbers such that $\mathrm{DEC}(a,b,k)$ holds. If $\lambda_4 = 0$ then $\mathrm{DEC}(a,b,k)$ simplifies to $\mathrm{DEC}_1(a,b)$ which is true. If $\lambda_4 > 0$, we take $P = k\lambda_4$ and we can see that $\mathrm{DEC}_2(a,b)$ is true.
($\Longleftarrow$) Assume first that $\mathrm{DEC}_1(a,b)$ is true. Then, taking $\lambda_4 = 0$ and $k = 0$ (any rational number would be fine for k), we see that $\exists k$. $\mathrm{DEC}(a,b,k)$ is true. Assume then that $\mathrm{DEC}_2(a,b)$ is true. Taking $k = P/\lambda_4$ (this is always possible as $\lambda_4 > 0$), we observe that there exists k such that $\mathrm{DEC}(a,b,k)$ is true. $\square$

For the positivity condition, we can prove in a similar way

Lemma 3.8. *Formula* $\exists k$. $\mathrm{POS}(a,b,k)$ *is equivalent to the disjunction* $\mathrm{POS}_1(a,b) \vee \mathrm{POS}_2(a,b)$.

In our case, $\mathrm{POS}_1(a,b)$ is equivalent to

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

and $\mathrm{POS}_2(a,b)$ to

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

Combining the previous results gives

Proposition 3.9. *Formula* $\exists k . \phi(a,b,k)$ *is equivalent to* $[\mathrm{DEC}_1(a,b) \vee \mathrm{DEC}_2(a,b)] \wedge [\mathrm{POS}_1(a,b) \vee \mathrm{POS}_2(a,b)]$.

Proof. Thanks to the previous lemmata, it only remains to justify the equivalence between the formulas $\exists k$. $\phi(a,b,k)$ and $\exists k$. $\mathrm{DEC}(a,b,k) \wedge \exists k$. $\mathrm{POS}(a,b,k)$.
($\Longrightarrow$) Let k_0 be a rational such that $\phi(a,b,k_0)$. We have $\mathrm{DEC}(a,b,k_0)$ and $\mathrm{POS}(a,b,k_0)$ because

$$\phi(a,b,k) \iff \mathrm{DEC}(a,b,k) \wedge \mathrm{POS}(a,b,k).$$

($\Longleftarrow$) Assume the existence of k_d such that $\mathrm{DEC}(a,b,k_d)$ and the existence of k_p such that $\mathrm{POS}(a,b,k_p)$. Then the rational $k_0 = \max(k_d, k_p)$ verifies $\mathrm{DEC}(a,b,k_0) \wedge \mathrm{POS}(a,b,k_0)$ and shows that $\exists k$. $\phi(a,b,k)$. $\square$

Back to our initial problem, the existence of an eventual linear ranking function is equivalent to the satisfiability of at least one of the following four linear systems:

$$\begin{aligned} &\mathrm{DEC}_1(a,b) \wedge \mathrm{POS}_1(a,b), \\ &\mathrm{DEC}_1(a,b) \wedge \mathrm{POS}_2(a,b), \\ &\mathrm{DEC}_2(a,b) \wedge \mathrm{POS}_1(a,b), \\ &\mathrm{DEC}_2(a,b) \wedge \mathrm{POS}_2(a,b), \end{aligned}$$

which we can decide in polynomial time. For our running example, $\mathrm{DEC}_2(a,b) \wedge \mathrm{POS}_1(a,b)$ is satisfiable as proved by the following SICStus Prolog query:

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

after compilation of the program:

```
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}.
```

The procedure we have informally outlined by means of examples is actually completely general. It is embodied in Algorithm 1, which is a (correct and complete) decision procedure for the existence of an eventual linear ranking function given a linear increasing function.

Algorithm 1 Existence of an eventual linear ranking function, given a linear increasing function

Require: C, an SLC loop $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}')$, and f, a linear increasing function for C
Ensure: Returns **true** if and only if, for some vector $\mathbf{a}$, $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$ is an eventual linear ranking function for (C, f).
$\mathrm{DEC}(\mathbf{a}, k) :=$ Farkas for the decreasing of ρ
$\mathrm{DEC}_1(\mathbf{a}), \mathrm{DEC}_2(\mathbf{a}) :=$ linearization of $\mathrm{DEC}(\mathbf{a}, k)$
$\mathrm{POS}(\mathbf{a}, k) :=$ Farkas for the positivity of ρ
$\mathrm{POS}_1(\mathbf{a}), \mathrm{POS}_2(\mathbf{a}) :=$ linearization of $\mathrm{POS}(\mathbf{a}, k)$
if $\bigvee_{1 \leq i,j \leq 2} \mathrm{DEC}_i(\mathbf{a}) \wedge \mathrm{POS}_j(\mathbf{a})$ is satisfiable **then**
  return true
else
  return false
end if

Theorem 3.10. *Let C be an SLC loop and f an increasing function for C.* Algorithm 1 *decides in polynomial time the existence of an eventual linear ranking function for* (C, f).

Computing an eventual linear ranking function ρ and its associated threshold k can be done as follows:

- if $\mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_1(\mathbf{a})$ is satisfiable, we compute a solution $\mathbf{a}$, $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$ is a standard linear ranking function and Proposition 3.6 applies;
- if $\mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_2(\mathbf{a})$ is satisfiable, we compute a solution $\mathbf{a}$, λ', P' and we take $k = P'/\lambda'_n$;
- if $\mathrm{DEC}_2(\mathbf{a}) \wedge \mathrm{POS}_1(\mathbf{a})$ is satisfiable, we compute a solution $\mathbf{a}$, λ, P and we take $k = P/\lambda_n$;

- if $\mathrm{DEC}_2(\mathbf{a}) \wedge \mathrm{POS}_2(\mathbf{a})$ is satisfiable, we compute a solution $\mathbf{a}$, λ, P, λ', P' and we take $k = \max(P/\lambda_n, P'/\lambda'_n)$.

Example 3.11. *Continuing with* Example 3.1*, here is the most general solution of* $\mathrm{DEC}_2(a, b) \wedge \mathrm{POS}_1(a, b)$*:*

```
?-  {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}.
?-
```

One particular solution is $b = 0 = \lambda_1 = \lambda_3 = \lambda'_2 = \lambda'_3$*,* $a = 1 = \lambda'_1 = \lambda_2 = \lambda_4$*,* $P = 1$*. Hence* $\rho(x, y) = x$ *is an eventual linear ranking function from the threshold* $k = P/\lambda_4 = 1$*.*

We also provide a decision procedure for the existence of an eventual *affine* ranking function.

Corollary 3.12. *The existence of an eventual affine ranking function for an SLC loop and associated increasing function,* (C, f)*, can be decided in polynomial time.*

Proof. From C, $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}')$, we construct C_a, $p(\mathbf{x}, y) \leftarrow c(\mathbf{x}, \mathbf{x}'),\ y = 1,\ y' = 1,\ p(\mathbf{x}', y')$, where y does not occur in $\mathbf{x}$. Note that C_a is an SLC loop and that $f_a(\mathbf{x}, y) = f(\mathbf{x})$ is an increasing function for C_a. Algorithm 1 applied to (C_a, f_a) gives an answer in polynomial time.

If Algorithm 1 returns **true** then, by correctness, there exists a threshold k and an eventual linear function $\rho_a(\mathbf{x}, y) = \langle \mathbf{a}, \mathbf{x} \rangle + by$ for (C_a, f_a). We readily check that $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle + b$ is an eventual affine ranking function for (C, f) from k.

If Algorithm 1 returns **false** then, by completeness, there is no eventual linear ranking function for (C_a, f_a). Assuming there exists an eventual affine ranking function $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle + b$ from k for (C, f), then $\rho_a(\mathbf{x}, y) = \langle \mathbf{a}, \mathbf{x} \rangle + by$ should be an eventual linear ranking function from k for (C_a, f_a), which is a contradiction. Hence there is no eventual affine ranking function for (C, f). $\square$

Example 3.13. *The SLC loop*

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

associated to the linear increasing function $f(x, y) = -y$ *does not admit an eventual linear ranking function, but does admit* $\rho(x, y) = x + 1$ *as an eventual affine ranking function from* $k = 1$*.*

3.2 Fully Automated Detection

We now consider the problem in its full generality: given an SLC loop C, does there exist an increasing function for C such that C admits an eventual linear ranking function?

Note that the space of increasing functions can be obtained as a convex set over their coefficients via the Farkas' Lemma and existentially quantified variables elimination.[3]

Definition 3.14. *Let* $C = \big(p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}')\big)$ *be an SLC loop. We denote by* INC *the set of vectors* $\mathbf{b}$ *such that* $f(\mathbf{x}) = \langle \mathbf{b}, \mathbf{x} \rangle$ *is increasing for* C*.*

[3]See also [6, Section 4.4].

Example 3.15. *A linear ranking function does not exist for the SLC loop* C

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

$\mathrm{INC} = \left\{ (b_1, b_2) \in \mathbb{Q} \times \mathbb{Q} \;\middle|\; b_1 \leq -2, b_1 - 2b_2 = 0 \right\}$ *induces the space of functions of the form* $f(x, y) = b_1 x + b_2 y$*, which are increasing for* C*.*

Let us consider the SLC loop of Example 3.15 associated to an increasing function $f(x, y) = b_1 x + b_2 y$ induced by INC. Defining $\rho(x, y) = a_1 x + a_2 y$ and considering b_1 and b_2 as parameters, ρ is an eventual linear ranking function when

$$\exists a_1, a_2, k \;.\; \forall x, y, x', y' : \begin{Bmatrix} x \geq 0, & x' \leq x + y, \\ b_1 x + b_2 y \geq k, & y' \leq -y - 1 \end{Bmatrix} \implies \begin{cases} a_1 x + a_2 y \geq 1 + a_1 x' + a_2 y', \\ a_1 x + a_2 y \geq 0. \end{cases}$$

This definition of the problem is denoted $\exists \mathbf{a}, k \;.\; \phi(\mathbf{a}, k)$. We can apply Farkas' Lemma as follows:

$$\begin{array}{rcccccccccl}
\lambda_1: & 1x & + & 0y & + & 0x' & + & 0y' & + 0 & \geq & 0 \\
\lambda_2: & 1x & + & 1y & - & 1x' & + & 0y' & + 0 & \geq & 0 \\
\lambda_3: & 0x & - & 1y & + & 0x' & - & 1y' & - 1 & \geq & 0 \\
\lambda\ : & b_1 x & + & b_2 y & + & 0x' & + & 0y' & - k & \geq & 0 \\
\implies & & & & & & & & & & \\
 & a_1 x & + & a_2 y & - & a_1 x' & - & a_2 y' & - 1 & \geq & 0 \\
 & a_1 x & + & a_2 y & & & & & & \geq & 0.
\end{array}$$

Formula $\phi(\mathbf{a}, k)$ is equivalent to the conjunction of formulas $\mathrm{DEC}(\mathbf{a}, k)$, i.e.,

$$\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, \lambda \geq 0 \;.\; \begin{cases} a_1 = \lambda_1 + \lambda_2 + b_1 \lambda & -a_1 = -\lambda_2, \\ a_2 = \lambda_2 - \lambda_3 + b_2 \lambda, & -a_2 = -\lambda_3, \quad -1 \geq -\lambda_3 - k\lambda, \end{cases}$$

ensuring the decreasing of the ranking function and $\mathrm{POS}(\mathbf{a}, k)$, that is,

$$\exists \lambda'_1 \geq 0, \lambda'_2 \geq 0, \lambda'_3 \geq 0, \lambda' \geq 0 \;.\; \begin{cases} a_1 = \lambda'_1 + \lambda'_2 + b_1 \lambda' & 0 = -\lambda'_2 \\ a_2 = \lambda'_2 - \lambda'_3 + b_2 \lambda', & 0 = -\lambda'_3 \quad 0 \geq -\lambda'_3 - k\lambda', \end{cases}$$

ensuring the positivity of the ranking function.

Let us focus on $\mathrm{DEC}(\mathbf{a}, k)$. We observe that the products with λ lead to a non-linearity that we can circumvent by noting that, as $\lambda \geq 0$, either $\lambda = 0$ or $\lambda > 0$. In the latter case, we introduce a vector $\mathbf{p} = (p_1, p_2)$ of two new variables where $p_1 = b_1 \lambda$ and $p_2 = b_2 \lambda$ together with, as previously, the new variable $P = k\lambda$. Formula $\exists k \;.\; \mathrm{DEC}(\mathbf{a}, k)$ is equivalent to the disjunction $\mathrm{DEC}_1(\mathbf{a}) \vee \exists \lambda, \mathbf{p}.\mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p})$ where in our case, $\mathrm{DEC}_1(\mathbf{a})$ is equivalent to

$$\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0 \;.\; \begin{cases} a_1 = \lambda_1 + \lambda_2, & -a_1 = -\lambda_2, \\ a_2 = \lambda_2 - \lambda_3, & -a_2 = -\lambda_3, \quad -1 \geq -\lambda_3, \end{cases}$$

and $\mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p})$ is equivalent to

$$\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, P\ . \begin{cases} a_1 = \lambda_1 + \lambda_2 + p_1, & -a_1 = -\lambda_2, \quad \lambda > 0, \\ a_2 = \lambda_2 - \lambda_3 + p_2, & -a_2 = -\lambda_3, \quad -1 \geq -\lambda_3 - P. \end{cases}$$

For the positivity condition, formula $\exists k\ .\ \mathrm{POS}(\mathbf{a}, k)$ is equivalent to the disjunction $\mathrm{POS}_1(\mathbf{a}) \vee \exists \lambda', \mathbf{p}'.\mathrm{POS}_2(\mathbf{a}, \mathbf{p}')$ where we introduce a vector $\mathbf{p}' = (p'_1, p'_2)$ of two new variables where $p'_1 = b_1\lambda'$, $p'_2 = b_2\lambda'$ together with, as previously, the new variable $P' = k\lambda'$. In our case, $\mathrm{POS}_1(\mathbf{a})$ is equivalent to

$$\exists \lambda'_1 \geq 0, \lambda'_2 \geq 0, \lambda'_3 \geq 0\ . \begin{cases} a_1 = \lambda'_1 + \lambda'_2, & 0 = -\lambda'_2, \\ a_2 = \lambda'_2 - \lambda'_3, & 0 = -\lambda'_3, \quad 0 \geq -\lambda'_3, \end{cases}$$

and $\mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}')$ to

$$\exists \lambda'_1 \geq 0, \lambda'_2 \geq 0, \lambda'_3 \geq 0, P'\ . \begin{cases} a_1 = \lambda'_1 + \lambda'_2 + p'_1, & 0 = -\lambda'_2, \quad \lambda' > 0, \\ a_2 = \lambda'_2 - \lambda'_3 + p'_2, & 0 = -\lambda'_3, \quad 0 \geq -\lambda'_3 - P'. \end{cases}$$

Back to our initial problem, the existence of an eventual linear ranking function is equivalent to the satisfiability of at least one of the following four systems:

1. $\mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_1(\mathbf{a})$: this case means that the increasing function and k are irrelevant. In other words, for each solution $\mathbf{a}$, $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$ is a standard linear ranking function and Proposition 3.6 applies.

2. $\mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}') \wedge \mathbf{p}'/\lambda' \in \mathrm{INC}$: note that satisfiability of $\mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}')$ is not sufficient, as its solution might lead to the coefficients $b_1 = p'_1/\lambda'$ and $b_2 = p'_2/\lambda'$ (λ' is strictly positive by definition), which could correspond to a non-increasing linear function. The third conjunct, $\mathbf{p}'/\lambda' \in \mathrm{INC}$, ensures that we stay within the space of increasing functions.

3. $\mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_1(\mathbf{a})$: this case is symmetric to previous one.

4. $\mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}') \wedge \mathbf{p}/\lambda = \mathbf{p}'/\lambda'$: this case combines the two previous ones. Note that the condition ensures that we consider the same linear ranking function and the same increasing function both in DEC_2 and in POS_2.

For our running example, the following SICStus Prolog query proves that $\mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_1(\mathbf{a})$ is satisfiable

```
?- dec2incpos1.
true.
?-
```

after compilation of the program

```
dec2incpos1 :-
   {% DEC2:
    L1 >= 0, L2 >= 0, L3 >= 0,
    A1 = L1 + L2 + P1, A1 = L2, L > 0,
    A2 = L2 - L3 + P2, A2 = L3, -1 >= -L3 - P,
    % INC: B1 =< -2, B1 - 2*B2 = 0
    P1 =< -2*L, P1 - 2*P2 = 0,
    % POS1:
    LP1 >= 0, LP2 >= 0, LP3 >= 0,
    A1 = LP1 + LP2, 0 = LP2,
    A2 = LP2 - LP3, 0 = LP3, 0 >= -LP3}.
```

The procedure we have informally outlined by means of examples is actually completely general and is embodied in Algorithm 2.

Algorithm 2 Existence of an eventual linear ranking function

Require: C, an SLC loop $p(\mathbf{x}) \leftarrow c(\mathbf{x}, \mathbf{x}'), p(\mathbf{x}')$
Ensure: Returns **true** if and only if there exists an increasing function f for C and $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$ such that ρ is an eventual linear ranking function for (C, f).
1: INC := the space of increasing functions for C
2: $\mathrm{DEC}(\mathbf{a}, k)$:= Farkas for the decreasing of ρ
3: $\mathrm{DEC}_1(\mathbf{a}), \mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p})$:= linearization of $\mathrm{DEC}(\mathbf{a}, k)$
4: $\mathrm{POS}(\mathbf{a}, k)$:= Farkas for the positivity of ρ
5: $\mathrm{POS}_1(\mathbf{a}), \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}')$:= linearization of $\mathrm{POS}(\mathbf{a}, k)$
6: $\phi_{1,1} := \mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_1(\mathbf{a})$
7: $\phi_{1,2} := \mathrm{DEC}_1(\mathbf{a}) \wedge \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}') \wedge \mathbf{p}'/\lambda' \in \mathrm{INC}$
8: $\phi_{2,1} := \mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_1(\mathbf{a})$
9: $\phi_{2,2} := \mathrm{DEC}_2(\mathbf{a}, \lambda, \mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_2(\mathbf{a}, \lambda', \mathbf{p}') \wedge \mathbf{p}/\lambda = \mathbf{p}'/\lambda'$
10: **if** $\bigvee_{1 \leq i,j \leq 2} \phi_{i,j}$ is satisfiable **then**
11:    **return true**
12: **else**
13:    **return false**
14: **end if**

THEOREM 3.16. *Let C be an SLC loop.* Algorithm 2 *decides the existence of an increasing function f and a linear function ρ such that ρ is an eventual linear ranking function for (C, f).*

Exactly as in the previous section, if Algorithm 2 returns **true** then we can extract an increasing function f, a threshold k, and a linear function ρ. We can also generalize the approach to the fully automated detection of eventual affine ranking functions.

With respect to complexity, Algorithm 2 is not polynomial for two reasons. In step 1, computing the set INC of linear increasing functions for C requires elimination of existentially quantified variables. In step 2, formula $\phi_{2,2}$ leads to a non-linear system and we may have to check its satisfiability in step 10. Although decidable, we are not aware of the existence of polynomial algorithms for these problems.

3.3 Verification

Given C an SLC loop, an associated increasing function f, and a linear function ρ, we want to know whether ρ is a ranking function. We can run Algorithm 1, with the coefficients $\mathbf{a}$ fully instantiated. If needed, we can compute the threshold k as explained in Section 3.1. It follows that the verification problem is polynomial.

3.4 Implementation

We have implemented both algorithms in SICStus Prolog. However, as $\phi_{2,2}$ of Algorithm 2 leads to a non-linear system, we relaxed this formula to

$$\mathrm{DEC}_2(\mathbf{a},\lambda,\mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_2(\mathbf{a},\lambda',\mathbf{p}') \wedge \mathbf{p}'/\lambda' \in \mathrm{INC},$$

which is now linear. As shown in the following proposition, the existence of an eventual linear ranking function (hence termination) is preserved, but the associated increasing function is not linear.

PROPOSITION 3.17. *Let C be an SLC loop and assume the truth of*

$$\mathrm{DEC}_2(\mathbf{a},\lambda,\mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC} \wedge \mathrm{POS}_2(\mathbf{a},\lambda',\mathbf{p}') \wedge \mathbf{p}'/\lambda' \in \mathrm{INC}.$$

Then there exists a non-linear increasing function f such that $\rho(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$ is an eventual linear ranking function for (C,f).

PROOF. As $\mathrm{DEC}_2(\mathbf{a},\lambda,\mathbf{p}) \wedge \mathbf{p}/\lambda \in \mathrm{INC}$ is true, there exists an increasing function f_d and a rational k_d such that when the value of f_d is beyond k_d, ρ decreases. Similarly, as $\mathrm{POS}_2(\mathbf{a},\lambda',\mathbf{p}') \wedge \mathbf{p}'/\lambda' \in \mathrm{INC}$ is true, there exists an increasing function f_p and a rational k_p such that when the value of f_p is beyond k_p, ρ is non-negative. Let $k = \max(k_p, k_d)$ and $f(\mathbf{x}) = \min\bigl(f_p(\mathbf{x}), f_d(\mathbf{x})\bigr)$. One readily checks that f is a non-linear increasing function for C and ρ is an eventual linear ranking function for (C, f). $\square$

4. RELATED WORK AND EXPERIMENTS

As eventual linear ranking functions generalize linear ranking functions, we focus on related work that goes beyond linear ranking functions for SLC loops. In order to appreciate the relative power of the different methods, we report on the results obtained with our algorithms on the loops discussed in the papers where the other approaches were introduced.

The method proposed in [27] repeatedly divides the state space to find a linear ranking function on each subspace, and then checks that the transitive closure of the transition relation is included in the union of the ranking relations. As the process may not terminate, one needs to bound the search. [27] also proposes a test suite, upon which we tested our approach. Our implementation analyzes the complete test suite in less than 2 seconds on a standard desktop computer. As expected, every loop [27, Table 1] which terminates with a linear ranking also has an eventual linear ranking. Moreover, loops 6, 12, 13, 18, 21, 23, 24, 26, 27, 28, 31, 32, 35, and 36 admit an eventual linear ranking function (which is discovered without using neither $\phi_{2,2}$ nor its relaxation). These are all shown terminating with the tool of [27]. On the other hand, loops 14, 34, and 38 do have a *disjunctive ranking function* (following the terminology of [27]), but do not admit an eventual linear ranking function.

[17] shows how to partition the loop relation into behaviors that terminate and behaviors to be analyzed in a subsequent termination proof after refinement. This work addresses both termination and conditional termination problems in the same framework. Concerning the benchmarks proposed in [17, Table 1], loops 6–41 all have an eventually linear ranking function except for loops 11, 14, 30, 34, and 38.

A method based on abstract interpretation for synthesizing ranking functions is described in [26]. Although the work contains no completeness result, the approach is able to discover piecewise-defined ranking functions.

Finally, let us point out that the concept of *eventual termination* appeared first in [13, 14]. The class loops studied in these works is wider but, as the technique of [14] relies on finite differences, this approach is incomplete. On the other hand, while [13] is also based on Farkas' Lemma, it seems [A. R. Bradley, Personal communication, May 2013] that the *polyranking* approach cannot prove, e.g., termination of the SLC loop $p(x, y) \leftarrow x \geq 1, x' = y, y' = y - 1, p(x', y')$, which admits an eventual linear ranking function.

5. CONCLUSION AND FUTURE WORK

We have proposed a definition of eventual linear ranking function for SLC loops that strictly generalizes the concept of linear ranking function. We also defined two correct and complete algorithms for detecting such ranking functions under different hypotheses. The first algorithm shows that the mere knowledge of the right increasing function allows checking the existence or even synthesizing an eventual linear ranking function in polynomial time. The second algorithm decides the existence of an eventual linear ranking function in its full generality but is not polynomial. We have also explained how to extend the algorithms for deciding eventual affine ranking functions. The algorithms admit a simple formulation as a constraint logic program and have been fully implemented in SICStus Prolog inside the BinTerm termination prover [24].

We plan to incorporate the algorithms in the Parma Polyhedra Library [4] and then in the ECLAIR static analyzer for C, C++ and Java programs.[4] This will enable us to conduct an extensive experimental evaluation on real programs. Moreover, as ECLAIR already includes linearization of floating-point constraints [8] as well as other sophisticated reasoning techniques over floating-point numbers (such as, e.g., [3]), the way is paved to explore a very interesting direction for future work: synthesis of eventual linear ranking functions for loops controlled by floating-point quantities.

It has to be noted that a nice property of the notion of eventual (not necessarily linear) ranking function is its simplicity. This is important when functions that witness termination have to be provided (and/or understood) by humans. This is the case when annotating a C/ACSL program with loop variants [7]: for the cases when a ranking function to be specified in a `loop variant` clause is not obvious, one could extend ACSL with a `loop prevariant` clause that allows the annotator to indicate a candidate increasing function. In the linear case, our first algorithm can efficiently decide whether the two clauses constitute a termination witness.

On the other hand, there obviously are, as indicated in Section 4, more complex classes of ranking functions and algorithms that allow to establish the termination of SLC loops that do not admit an eventual linear ranking functions. A proper assessment of the relative merits of these approaches, all extremely recent, requires an extensive experimental evaluation that, as mentioned already, is one of the main objectives for future work.

The verification of linear ranking functions for integer SLC loops, that is to say, checking the satisfiability of

$$c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 1 + \rho(\mathbf{x}')$$

and

$$c(\mathbf{x}, \mathbf{x}') \wedge \rho(\mathbf{x}) < 0,$$

[4] `http://bugseng.com/products/eclair`

is an NP-complete problem. Concerning the existence of linear ranking functions, as the Farkas' Lemma is not true for the integers, the method presented in Section 2 is not valid. The problem, which has been solved very recently in [10], is coNP-complete, and the paper proposes an exponential-time algorithm. Extending the present approach to integer SLC loops is another interesting idea to consider for future work.

Acknowledgments.
We are grateful to Anthony Alezan, Aaron R. Bradley, Étienne Payet, and some anonymous referees for their helpful comments.

6. REFERENCES

[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] A. Alezan, R. Bagnara, F. Mesnard, and E. Payet. Détection des fonctions de rang linéaires à terme. In *Actes des NeuviÃĺmes JournÃľes Francophones de Programmation par Contraintes (JFPC 2013)*, pages 11–20, Aix-en-Provence, France, 2013. In French.

[3] R. Bagnara, M. Carlier, R. Gori, and A. Gotlieb. Symbolic path-oriented test data generation for floating-point programs. In *Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation*, Luxembourg City, Luxembourg, 2013. IEEE Press.

[4] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[5] R. Bagnara, F. Mesnard, A. Pescetti, and E. Zaffanella. The automatic synthesis of linear ranking functions: The complete unabridged version. Report `arXiv:cs.PL/1004.0944v2`, 2012. Available at `http://arxiv.org/` and `http://bugseng.com/products/ppl/`.

[6] 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.

[7] P. Baudin, P. Cuoq, J.-C. Filliâtre, C. Marché, B. Monate, Y. Moy, and V. Prevosto. *ACSL: ANSI/ISO C Specification Language*. CEA LIST and INRIA, 1.7 edition, 2013.

[8] M. S. Belaid, C. Michel, and M. Rueher. Boosting local consistency algorithms over floating-point numbers. In M. Milano, editor, *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 127–140, Québec City, Canada, 2012. Springer-Verlag, Berlin.

[9] 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/`.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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)*, volume 4144 of *Lecture Notes in Computer Science*, pages 372–385, Seattle, WA, USA, 2006. Springer.

[16] 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.

[17] P. Ganty and S. Genaim. Proving termination starting from the end. Technical Report `arXiv:abs/1302.4539`, 2013. Preprint version [18].

[18] P. Ganty and S. Genaim. Proving termination starting from the end. In N. Sharygina and H. Veith, editors, *Computer Aided Verification: Proceedings of the 25th International Conference (CAV 2013)*, volume 8044 of *Lecture Notes in Computer Science*, pages 397–412, Saint Petersburg, Russia, 2013. Springer.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] C. Urban. The abstract domain of segmented ranking functions. In F. Logozzo and M. Fahndrich, editors, *Proceedings of the 20th International Symposium on Static Analysis (SAS 2013)*, volume 7935 of *Lecture Notes in Computer Science*, pages 43–62, Seattle, WA, USA, 2013. Springer.

[27] H. Yi Chen, S. Flur, and S. Mukhopadhyay. Termination proofs for linear simple loops. In A. Miné and D. Schmidt, editors, *Proceedings of the 19th International Symposium on Static Analysis (SAS 2012)*, volume 7460 of *Lecture Notes in Computer Science*, pages 422–438, Deauville, France, 2012. Springer.