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 \( \mathbb{Q} \).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

Copyright is held by the owner/author(s). Publication rights licensed to ACM. Copyright 2013 ACM 978-1-4503-2154-9/13/09 ...$15.00 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 \( \mathbf{v}^T \). For each \( i = 1, \ldots, n \), \( v_i \) denotes the \( i \)-th component of the (column) vector \( \mathbf{v} = (v_1, \ldots, v_n)^T \in \mathbb{Q}^n \). The scalar product of \( \mathbf{v}, \mathbf{w} \in \mathbb{Q}^n \), denoted \( \langle \mathbf{v}, \mathbf{w} \rangle \), is the rational number

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

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

\[
\text{while } (B \ x \leq \ b) \ \text{ do } A \left( \frac{x}{x'} \right) \leq c
\]

where \( x = (x_1, \ldots, x_n)^T \) and \( x' = (x_1', \ldots, x_n')^T \) are column vectors of variables, \( B \in \mathbb{Z}^{p \times n} \) is an integer matrix, \( b \in \mathbb{Z}^p \), \( A \in \mathbb{Z}^{q \times 2n} \) and \( c \in \mathbb{Z}^q \).

Such a loop can be conveniently written as a constraint logic programming rule:

\[
p(x) \leftarrow Bx \leq b, \ A \left( \frac{x}{x'} \right) \leq c, \ p(x').
\]

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 \( x \); if \( Bx \leq b \) is false, the loop terminates; otherwise, a new point \( x' \) is chosen that satisfies

\[
A \left( \frac{x}{x'} \right) \leq c
\]

and iteration continues replacing the values of \( x \) by the corresponding ones of \( x' \).

Loop termination can always be ensured by a ranking function \( \rho \), a function from \( \mathbb{Z}^n \) or \( \mathbb{Q}^n \) to a well-founded set. As the range of \( \rho \) 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 \( x' := Ax \), 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. 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

\[ p(x) \leftarrow c(x, x'), p(x') \]

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

\[ \forall x, x': c(x, x') \implies \rho(x) \geq 1 + \rho(x') \land \rho(x) \geq 0. \]

In words, the continuation of the iteration, i.e., \( c(x, x') \), entails that \( p \) stays positive and strictly decreases by at least 1 for each iteration. We point out that if \( c(x, x') \) is not satisfiable, the loop ends immediately and any linear function is a ranking function. In the paper, we assume that \( c(x, x') \) 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 \( \varepsilon \), a fixed positive quantity. Actually, by multiplying such an \( \varepsilon \)-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

\[ p(x) \leftarrow c(x, x'), p(x') \]

is a linear ranking function for

\[ p(x, y) \leftarrow c(x, x'), y = 1, y' = 1, p(x', y') \]

where \( y \) is distinct from the variables in \( x \).

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.

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

\[
\begin{align*}
& a_1 x_1 + \cdots + a_n x_n + b_1 \geq 0 \\
& \ldots + a_1 x_1 + \cdots + a_n x_n + b_m \geq 0.
\end{align*}
\]

and suppose that \( S \) has at least one solution. Farkas’ Lemma states the equivalence of

\[ \forall x_1, \ldots, x_n : S \implies (c_1 x_1 + \cdots + c_n x_n + d \geq 0) \]

and

\[ \exists \lambda_1 \geq 0, \ldots, \lambda_m \geq 0 . \]

\[ (d \geq \sum_{i=1}^{m} \lambda_i b_i) \land \bigwedge_{j=1}^{n} (c_j = \sum_{i=1}^{m} \lambda_i a_{i,j}). \]

2.2 Verification

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

\[ c(x, x') \land p(x) < 1 + \rho(x') \]

or

\[ c(x, x') \land p(x) < 0. \]

This test has polynomial complexity and can be done with a complete rational solver such as, e.g., CLP(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 \( p(x, y) = x \) is a ranking function, as proved by the following SICStus Prolog session.

?- use_module(library(clpq)).
% library(clpq) compiled
true.
?- \{X \geq 0, Y1 \leq Y - 1, X1 \leq X + Y, Y \leq -1, X < 1 + X1\}.
false.
?- \{X \geq 0, Y1 \leq Y - 1, X1 \leq X + Y, Y \leq -1, X < 0\}.
false.
?- 

2.3 Detection

Given an SLC loop, we would like to know whether it admits a linear ranking function \( \rho \). 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 \( \rho(x, y) = ax + by \):

\[
\exists a, b. \forall x, y, x', y' : \begin{cases} x \geq 0, & x' \leq x + y, \\ y \leq -1, & y' \leq y - 1 \end{cases} \implies \begin{cases} ax + by \geq 1 + ax' + by', \\ ax + by \geq 0. \end{cases}
\]
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 x<=x+y and y<=-1)
   impl (a*x+b*y>=1+a*x1+b*y1 and a*x+b*y>=0)));
4: rlqe F;
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
\[ a^2 - ab \geq 0 \land a - b \neq 0 \land a > 0 \land b = 0 \]
\[ \land (a^2 - ab^2 \leq 0 \lor a^2 - ab = 0 \lor a^2 - 2ab - a + b^2 + b \geq 0) \]
\[ \land (a^2 - ab = 0 \lor a^2 - 2ab - a + b^2 \geq 0), \]
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]. Considering \( a \) and \( b \) as parameters of the problem, we can apply Farkas’ Lemma. For the strict decrease of the ranking function we have
\[ \forall x, y, x', y' : \left\{ \begin{array}{l}
   x \geq 0, \quad x' \leq x + y, \\
   y \leq -1, \quad y' \leq y - 1
\end{array} \right. \]
\[ \impl \ ax + by \geq 1 + ax' + by'. \] (2)

Application of Farkas’ Lemma to this problem can be depicted as follows:
\[ \lambda_1 : 1x + 0y + 0x' + 0y' + 0 \geq 0 \]
\[ \lambda_2 : 1x + 1y - 1x' + 0y' + 0 \geq 0 \]
\[ \lambda_3 : 0x + 1y + 0x' - 1y' - 1 \geq 0 \]
\[ \lambda_4 : 0x - 1y + 0x' + 0y' - 1 \geq 0 \]
\[ \impl \ ax + by - ax' - by' - 1 \geq 0 \]

We know that formula (2) is equivalent to the existence of four non-negative rational numbers \( \lambda_1, \ldots, \lambda_4 \) such that:
\[ \begin{cases}
   a = \lambda_1 + \lambda_2, \\
   b = \lambda_2 + \lambda_3 - \lambda_4, \\
   -a = -\lambda_2, \\
   -b = -\lambda_3 - \lambda_4.
\end{cases} \] (3)

The positivity of the ranking function, that is,
\[ \forall x, y, x', y' : \left\{ \begin{array}{l}
   x \geq 0, \quad x' \leq x + y, \\
   y \leq -1, \quad y' \leq y - 1
\end{array} \right. \]
\[ \impl \ ax + by \geq 0 \] (4)
can be written as
\[ \begin{align*}
   \lambda_1' : & \quad 1x + 0y + 0x' + 0y' + 0 \geq 0 \\
   \lambda_2' : & \quad 1x + 1y - 1x' + 0y' + 0 \geq 0 \\
   \lambda_3' : & \quad 0x + 1y + 0x' - 1y' - 1 \geq 0 \\
   \lambda_4' : & \quad 0x - 1y + 0x' + 0y' - 1 \geq 0
\end{align*} \]
\[ \impl \ ax + by + 0x' + 0y' + 0 \geq 0. \]

By Farkas’ Lemma, formula (4) is equivalent to the existence of four other non-negative rational numbers \( \lambda_1', \ldots, \lambda_4' \) such that:
\[ \begin{cases}
   a = \lambda_1' + \lambda_2', \\
   b = \lambda_2' + \lambda_3' - \lambda_4', \\
   0 = -\lambda_2', \\
   0 \geq -\lambda_3' - \lambda_4'.
\end{cases} \] (5)

Summarizing, by Farkas Lemma, formula (1) is equivalent to the conjunction of formulas (3) and (5):
\[ \exists a, b . \ \exists \lambda_1, \ldots, \lambda_4, \lambda_1', \ldots, \lambda_4' \geq 0 . \]
\[ \begin{cases}
   a = \lambda_1 + \lambda_2, \\
   b = \lambda_2 + \lambda_3 - \lambda_4, \\
   -a = -\lambda_2, \\
   -b = -\lambda_3 - \lambda_4.
\end{cases} \] (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 \( \lambda_1 \) and \( \lambda_1' \) from (6) using, e.g., the algorithm of Fourier-Motzkin. For example the SICStus Prolog program
\[ \text{fm}(A, B) :- \]
\[ \{ L1 \geq 0, \quad L2 \geq 0, \quad L3 \geq 0, \quad L4 \geq 0, \]
\[ \text{LP1} \geq 0, \quad \text{LP2} \geq 0, \quad \text{LP3} \geq 0, \quad \text{LP4} \geq 0, \]
\[ A = L1 + L2, \quad B = L2 + L3 - L4, \]
\[ A = L2, \quad B = L3, \quad 1 \approx L3 + L4, \]
\[ A = \text{LP1} + \text{LP2}, \quad B = \text{LP2} + \text{LP3} - \text{LP4}, \]
\[ 0 = \text{LP2}, \quad 0 = \text{LP3}, \quad 0 = \text{LP3} + \text{LP4} \} \]
can be queried as follows:
\[ | \text{?- fm}(A, B). \]
\[ B = 0, \quad \{ A \geq 1 \}. \]
\[ | \text{?-} \]

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
\[ p(x, y) \rightarrow x \geq 0, y' \leq y - 1, x' \leq x + y, p(x', y') \]
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 \( f(x) \) that increases at each iteration of the loop in the following sense:

**Definition 3.2.** Let \( C \) be the SLC loop
\[ p(x) \leftarrow c(x, x'), p(x'). \]
A function \( f(x) \) is increasing for \( C \) if it is linear and satisfies: \( \forall x, x' : c(x, x') \implies f(x') \geq 1 + f(x) \).

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
\[ p(x) \leftarrow c(x, x'), p(x'). \]
where \( p \) is an \( n \)-ary relation; let also \( f(x) \) be a linear increasing function for \( C \). An eventual linear ranking function \( \rho \) for \( (C, f) \) is a linear map of \( \mathbb{Q}^n \) to \( \mathbb{Q} \) such that
\[ \exists k : \forall x, x' : (c(x, x') \land f(x) \geq k) \implies (\rho(x) \geq 1 + \rho(x') \land \rho(x) \geq 0). \]

For comparison with Definition 2.1, remark that the threshold \( k \) is existentially quantified and that \( f(x) \geq k \) 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 \( \rho \) 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 \( \rho \) is a linear ranking function for \( C \), then there exists an increasing function \( s \) such that \( (C, f) \) has an eventual linear ranking function.

**Proof.** By hypothesis, there exists a linear ranking function \( \rho(x) \) for \( C \). The linear function \( f(x) = \rho(x) \) is non-positive and strictly increasing for \( C \). Considering \( k = 1 \) it can be seen that the function \( \rho'(x) = 0 \) is an eventual linear ranking function for \( (C, f) \).

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 \( f(x, y) = ax + by \), \( \rho \) is an eventual linear ranking function when
\[ \exists a, b, k : \forall x, y, x', y' : \begin{cases} x \geq 0, & x' \leq x + y, \\ y \geq k, & y' \leq y - 1 \end{cases} \implies \begin{cases} ax + by \geq 1 + ax' + by', \\ ax + by \geq 0. \end{cases} \]

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:
\[
\begin{align*}
\lambda_1 & : 1x + 0y + 0x' + 0y' + 0 \geq 0 \\
\lambda_2 & : 1x + 1y - 1x' + 0y' + 0 \geq 0 \\
\lambda_3 & : 0x + 1y + 0x' - 1y' - 1 \geq 0 \\
\lambda_4 & : 0x - 1y + 0x' + 0y' - k \geq 0 \\
\end{align*}
\]

Hence, formula \( \phi(a, b, k) \) is equivalent to the conjunction of formulas \( \text{DEC}(a, b, k) \), 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 \( \text{POS}(a, b, k) \), 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 \( \text{DEC}(a, b, k) \). 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:
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 \( \rho \) and its associated threshold \( k \) can be done as follows:

- if \( \text{DEC}(a, k) \land \text{POS}_1(a, k) \) is satisfiable, we compute a solution \( a, \rho(x) = \langle a, x \rangle \) is a standard linear ranking function and Proposition 3.6 applies;
- if \( \text{DEC}(a, k) \land \text{POS}_2(a, k) \) is satisfiable, we compute a solution \( a, X^* \) and we take \( k = P/L(X^*) \);
- if \( \text{DEC}(a, k) \land \text{POS}(a, k) \) is satisfiable, we compute a solution \( a, \lambda, P \) and we take \( k = P/L(X^*) \).
if \(\text{DEC}(a) \wedge \text{POS}(a)\) is satisfiable, we compute a solution \(a, \lambda, P, x', P'\) and we take \(k = \max(P/\lambda a_{\text{a}}, P'/\lambda c)\).

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

\[
\L_1 \geq 0, \quad L_2 \geq 0, \quad L_3 \geq 0, \quad L_4 > 0
\]

The most general solution of \(\text{DEC}\) is no eventual linear ranking function for \((a, b, x, y)\). Assumptions \(a, b \geq 0\) induce 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, \(\rho\) is an eventual linear ranking function when

\[
\exists a_1, a_2, k. \forall x, y, x', y' : \left\{ \begin{array}{l}
x \geq 0, \quad x' \leq x + y, \\
b_1 x + b_2 y \geq k, \quad y' \leq -y - 1
\end{array} \right. \implies \left\{ \begin{array}{l}
a_1 x + a_2 y \geq 1 + a_1 x' + a_2 y', \\
a_1 x + a_2 y' \geq 0.
\end{array} \right.
\]

This definition of the problem is denoted \(\exists a, k . \phi(a, k)\). We can apply Farkas’ Lemma as follows:

\[
\begin{align*}
\lambda_1 & : \quad 1 x + 0 y + 0 x' + 0 y' + 0 \geq 0 \\
\lambda_2 & : \quad 1 x + 1 y - 1 x' + 0 y' + 0 \geq 0 \\
\lambda_3 & : \quad 0 x - 1 y + 0 x' - 1 y' - 1 \geq 0 \\
\lambda & : \quad b_1 x + b_2 y + 0 x' + 0 y' - k \geq 0
\end{align*}
\]

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

\[
\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, \lambda \geq 0.
\]

\[
\begin{align*}
a_1 & = \lambda_1 + \lambda_2 + b_1 \lambda - a_1 = -\lambda_2, \\
a_2 & = \lambda_2 - \lambda_3 + b_2 \lambda, -a_2 = -\lambda_3, -1 \geq -\lambda_3 - k \lambda.
\end{align*}
\]

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

\[
\exists \lambda_1' \geq 0, \lambda_2' \geq 0, \lambda_3' \geq 0, \lambda' \geq 0.
\]

\[
\begin{align*}
a_1 & = \lambda_1' + \lambda_2' + b_1 \lambda' - a_1 = -\lambda_2', \\
a_2 & = \lambda_2' - \lambda_3' + b_2 \lambda', 0 = -\lambda_3' \geq -\lambda_3 - k \lambda',
\end{align*}
\]

ensuring the positivity of the ranking function.

Let us focus on \(\text{DEC}(a, k)\). We observe that the products with \(\lambda\) 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 \(p = (p_1, p_2)\) 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 3.38: \(\text{DEC}(a, k)\) is equivalent to the disjunction \(\text{DEC}(a) \vee 3.38.\ a . \text{DEC}(a, \lambda, P)\) where in our case, \(\text{DEC}(a)\) is equivalent to

\[
\exists \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0.
\]

\[
\begin{align*}
a_1 & = \lambda_1 + \lambda_2, -a_1 = -\lambda_2, \\
a_2 & = \lambda_2 - \lambda_3, -a_2 = -\lambda_3, -1 \geq -\lambda_3,
\end{align*}
\]

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

3See also [6, Section 4.4].
and DEC\(_2(a, \lambda, p)\) is equivalent to
\[\exists \lambda_1 \geq 0, \lambda_2 \geq 0, P.\]
\[
\begin{align*}
& a_1 = \lambda_1 + \lambda_2 + p_1, \quad -a_1 = -\lambda_2, \quad \lambda > 0, \\
& a_2 = \lambda_2 - \lambda_3 + p_2, \quad -a_2 = -\lambda_3, \quad -1 \geq -\lambda_3 - P.
\end{align*}
\]
For the positivity condition, formula \(\exists k \cdot \text{POS}(a, k)\) is equivalent to the disjunction \(\text{POS}_1(a) \lor 3 \lambda', p'.\text{POS}_2(a, p')\) where we introduce a vector \(p' = (p_1, p_2)\) of two new variables where \(p_1' = b_1 X', p_2' = b_2 X'\) together with, as previously, the new variable \(P' = k X'\). In our case, \(\text{POS}_1(a)\) is equivalent to
\[\exists \lambda_1' \geq 0, \lambda_2' \geq 0, \lambda_3' \geq 0, \]
\[
\begin{align*}
& a_1 = \lambda_1' + \lambda_2', \quad 0 = -\lambda_2', \quad \lambda' > 0, \\
& a_2 = \lambda_2' - \lambda_3' + p_2, \quad 0 = -\lambda_3', \quad 0 \geq -\lambda_3' - P'.
\end{align*}
\]
and \(\text{POS}_2(a, \lambda', p')\) to
\[\exists \lambda_1' \geq 0, \lambda_2' \geq 0, \lambda_3' \geq 0, P'.
\]
\[
\begin{align*}
& a_1 = \lambda_1' + \lambda_2', \quad 0 = -\lambda_2', \quad \lambda' > 0, \\
& a_2 = \lambda_2' - \lambda_3' + p_2, \quad 0 = -\lambda_3', \quad 0 \geq -\lambda_3' - P'.
\end{align*}
\]
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. DEC\(_1(a) \land \text{POS}_1(a)\): this case means that the increasing function and \(k\) are irrelevant. In other words, for each solution \(a, \rho(x) = (a, x)\) is a standard linear ranking function and Proposition 3.6 applies.

2. DEC\(_1(a) \land \text{POS}_2(a, \lambda', p')\) \(\land p'/\lambda' \in \text{INC}\): note that satisfiability of DEC\(_1(a) \land \text{POS}_2(a, \lambda', p')\) is not sufficient, as its solution might lead to the coefficients \(b_1 = p_1'/X'\) and \(b_2 = p_2'/X'\) (\(X'\) is strictly positive by definition), which could correspond to a non-increasing linear function. The third conjunct, \(p'/\lambda' \in \text{INC}\), ensures that we stay within the space of increasing functions.

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

4. DEC\(_2(a, \lambda, p)\) \(\land p/\lambda \in \text{INC} \land \text{POS}_2(a, \lambda', p')\) \(\land p'/\lambda' = p'/\lambda'\): 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 DEC\(_2(a, \lambda, p)\) \(\land p/\lambda \in \text{INC} \land \text{POS}_1(a)\) is satisfiable.

?- dec2incpos1.
true.
?- after compilation of the program

\[
\text{dec2incpos1} :-
\begin{align*}
& \% \text{DEC}_2: \\
& L_1 \geq 0, \quad L_2 \geq 0, \quad L_3 \geq 0, \\
& A_1 = L_1 + L_2 + P_1, \quad A_1 = L_2, \quad L > 0, \\
& A_2 = L_2 - L_3 + P_2, \quad A_2 = L_3, \quad -1 \geq -L_3 - P, \\
\end{align*}
\]

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 \(p(x) = c(x, x'), p(x')\)

**Ensure:** Returns true if and only if there exists an increasing function \(\rho\) for \(C\) such that \(\rho\) is an eventual linear ranking function for \((C, f)\).

1: INC := the space of increasing functions for \(C\)
2: DEC\(_4(a, k) := \text{Farkas for the positivity of} \rho\)
3: DEC\(_1(a), \text{DEC}_2(a, \lambda, p) := \text{linearization of DEC}_1(a, k)\)
4: \(\text{POS}_1(a, k) := \) Farkas for the positivity of \(\rho\)
5: \(\text{POS}_1(a), \text{POS}_2(a, \lambda', p') := \text{linearization of POS}_1(a)\)
6: \(\phi_{i+1} := \text{DEC}_1(a) \land \text{POS}_1(a)\)
7: \(\phi_{i+2} := \) DEC\(_3(a, \lambda, p) \land p/\lambda \in \text{INC} \land \text{POS}_1(a)\)
8: \(\phi_{i+2} := \) DEC\(_2(a, \lambda, p) \land p/\lambda \in \text{INC} \land \text{POS}_2(a, \lambda', p')\)
9: \(\phi_{i+2} := \) DEC\(_3(a, \lambda, p) \land p/\lambda = p'/\lambda'\)
10: if \(\bigwedge_{1 \leq i \leq k} \phi_{i+2}\) is satisfiable then
11: \quad return true
12: else
13: \quad return false
14: endif

**Theorem 3.16:** Let \(C\) be an SLC loop. Algorithm 2 decides the existence of an increasing function \(f\) and a linear function \(\rho\) such that \(\rho\) 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 \(\rho\). 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 \(\phi_{i+2}\) 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 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 \(\phi_{i+2}\) of Algorithm 2 leads to a non-linear system,
we relaxed this formula to
\[ \text{DEC}_2(a, \lambda, \rho) \land p/\lambda \in \text{INC} \land \text{POS}_2(a, \lambda', \rho') \land p'/\lambda' \in \text{INC}, \]
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
\[ \text{DEC}_2(a, \lambda, \rho) \land p/\lambda \in \text{INC} \land \text{POS}_2(a, \lambda', \rho') \land p'/\lambda' \in \text{INC}. \]
Then there exists a non-linear increasing function \( f \) such that \( \rho(x) = (a, x) \) is an eventual linear ranking function for \((C, f)\).

**Proof.** As \( \text{DEC}_2(a, \lambda, \rho) \land p/\lambda \in \text{INC} \) is true, there exists an increasing function \( f_2 \) and a rational \( k_2 \) such that when the value of \( f_2 \) is beyond \( k_2 \), \( \rho \) decreases. Similarly, as \( \text{POS}_2(a, \lambda', \rho') \land p'/\lambda' \in \text{INC} \) is true, there exists an increasing function \( f_3 \) and a rational \( k_3 \) such that when the value of \( f_3 \) is beyond \( k_3 \), \( \rho \) is non-negative. Let \( k = \max(k_2, k_3) \) and \( f(x) = \min(f_2(x), f_3(x)) \). One readily checks that \( f \) is a non-linear increasing function for \( C \) and \( \rho \) is an eventual linear ranking function for \((C, f)\). □

### 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 \( p(x, y) \rightarrow x \geq 1, x' = y, y' = y-1, p(x', y') \), 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. 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 invariant 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
\[ c(x, x') \land \rho(x) < 1 + \rho(x') \]
and
\[ c(x, x') \land \rho(x) < 0, \]

4http://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


