On the Linear Ranking Problem for Integer Linear-Constraint Loops

Amir M. Ben-Amram
The Tel-Aviv Academic College
amirben@mta.ac.il

Samir Genaim
Complutense University of Madrid
samir.genaim@fdi.ucm.es

Abstract

In this paper we study the complexity of the Linear Ranking problem: given a loop, described by linear constraints over a finite set of *integer variables*, is there a linear ranking function for this loop? While existence of such a function implies termination, this problem is not equivalent to termination. When the variables range over the rationals or reals, the Linear Ranking problem is known to be PTIME decidable. However, when they range over the integers, whether for single-path or multipath loops, the complexity of the Linear Ranking problem has not yet been determined. We show that it is coNP-complete. However, we point out some special cases of importance of PTIME complexity. We also present complete algorithms for synthesizing linear ranking functions, both for the general case and the special PTIME cases.

Categories and Subject Descriptors F.2.0 [*Analysis of Algorithms and Problem Complexity*]: General; F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs

General Terms Verification, Theory.

Keywords Ranking Functions, Termination, Linear Constraints.

1. Introduction

Termination analysis has received a considerable attention and nowadays several powerful tools for the automatic termination analysis of different programming languages and computational models exist [1, 21, 27, 43]. Much of the recent development in termination analysis has benefited from techniques that deal with one loop at a time, where a loop is specified by a loop guard and a (non-iterative) loop body.

Very often, these loops are abstracted so that the state of the program during the loop is represented by a finite set of integer variables, the loop guard is a conjunction of linear inequalities, and the body modifies the variables in an affine linear way, as in the following example:

$$\begin{array}{l} \textit{while } (x_2 - x_1 \le 0, x_1 + x_2 \ge 1) \textit{ do} \\ \qquad\qquad x_2' = x_2 - 2x_1 + 1, x_1' = x_1 \end{array} \tag{1}$$

where tagged variables represent the values at the completion of an iteration. When the variables are modified so that they are not affine linear functions of the old ones, the effect is sometimes captured (or approximated) by means of *linear constraints*. E.g., the C language loop "`while (4*x>=y && y>=1) do x=(2*x+1)/5;`", which involves integer division, can be represented by linear constraints as follows (since `2*x+1` is always positive)

$$\begin{array}{l} \textit{while } (4x_1 \ge x_2, x_2 \ge 1) \textit{ do} \\ \qquad -2x_1 + 5x_1' \le 1, 2x_1 - 5x_1' \le 3, x_2' = x_2 \end{array} \tag{2}$$

Linear constraints might also be used to model changes to data structures, the variables representing a size abstraction such as length of lists, depth of trees, etc. [17, 34, 35, 43]. For a precise definition of the loop representations we consider, see Sec. 2; they also include *multipath loops* where alternative paths in the loop body are represented.

A standard technique to prove the termination of a loop is to find a ranking function. Such a function maps a program state (a valuation of the variables) into an element of some well-founded ordered set, such that the value descends (in the appropriate order) whenever the loop completes an iteration. Since descent in a well-founded set cannot be infinite, this proves that the loop must terminate. This definition of "ranking function" is very general; in practice, researchers have often limited themselves to a convenient and tractable form of ranking function, so that an algorithm to find the function—if there is one—might be found.

A frequently used class of ranking functions is based on *affine linear functions*. In this case, we seek a function $\rho(x_1, \ldots, x_n) = a_1x_1 + \cdots + a_nx_n + a_0$, with the rationals as a co-domain, such that (i) $\rho(\bar{x}) \ge 0$ for any valuation $\bar{x}$ that satisfies the loop guard; and (ii) $\rho(\bar{x}) - \rho(\bar{x}') \ge 1$ for any transition that starts in $\bar{x}$ and continues in $\bar{x}'$. This automatically induces the piecewise-linear ranking function: $f(\bar{x}) = \rho(\bar{x}) + 1$ if $\bar{x}$ satisfies the loop guard and 0 otherwise, with the non-negative rationals as a co-domain but ordered w.r.t. $a \preceq b$ iff $a + 1 \le b$ (which is well-founded). For simplicity, we call ρ itself a *linear ranking function* instead of referring to f.

An algorithm to find a linear ranking function using linear programming (LP) was found by multiple researchers in different places and times and in some alternative versions [3, 20, 25, 36, 38, 42]. Since LP has a polynomial time complexity, most of these methods yield polynomial-time algorithms. Generally speaking, they are based on the fact that LP can precisely decide whether a given inequality is implied by a set of other inequalities, and can even be used to generate any implied inequality. After all, conditions (i) and (ii) above are inequalities that should be implied by the constraints that define the loop guard and body. This approach can, in a certain sense, be *sound and complete*

Soundness means that it produces a correct linear ranking function, if it succeeds; completeness means that if a linear ranking function exists, it will succeed. In other words, there are no *false negatives*. A completeness claim appears in some of the references,

POPL'13, January 23–25, 2012, Rome, Italy.

and we found it cited several times. In our opinion, it has created a false impression that the Linear Ranking problem for linear-constraint loops with *integer variables* was completely solved (and happily classified as polynomial time).

The fly in the ointment is the fact that these solutions are only complete when the variables range *over the rationals*, which means that the linear ranking function has to fulfill its requirements for any rational valuation of the variables that satisfies the loop guard. But this may lead to a false negative if the variables are, in fact, integers. The reader may turn to the two loops above and note that both of them do not terminate over the rationals at all (for the first, consider $x_1 = x_2 = \frac{1}{2}$; for the second, $x_1 = \frac{1}{4}$ and $x_2 = 1$). But they have linear ranking functions valid for all integer valuations, which we derive in Sec. 3.4.

This observation has led us to investigate the Linear Ranking problem for single-path and multipath linear constraint loops. We present several fundamental new results on this problem. We have confirmed that this problem is indeed harder in the integer setting, proving it to be coNP-complete (as a decision problem), even for loops that only manipulate integers in a finite range. On a positive note, this shows that there *is* a complete solution, even if exponential-time. We give such a solution both to the decision problem and to the synthesis problem. The crux of the coNP-completeness proof, and the corresponding synthesis algorithm, is that we rely on the *generator representation* of the transition polyhedron defined by the loop constraints. We provide sufficient and necessary conditions for the existence of a linear ranking function that use the vertices and rays of this representation.

Another positive news for the practically-minded reader is that some special cases of importance do have a PTIME solution, because they reduce (with no effort, or with a polynomial-time computation) to the rational case. We present several such cases, which include, among others, loops in which the body is a sequence of linear affine updates with integer coefficients, as in loop (1) above; and the condition is defined by either an extended form of *difference constraints*, a restricted form of *Two Variables Per Inequality* constraints, or a cone (constraints where the free constant is zero). Some cases in which the body involves linear constraints are also presented. All the algorithms presented in this paper have been implemented, and can be tried out online (see Sec. 5).

Our results should be of interest to all users of linear ranking functions, and in fact their uses go beyond termination proofs. For example, they have been used to provide an upper bound on the number of iterations of a loop in *program complexity analysis* [2, 3] and to automatically parallelize computations [24, 25]. We remark that in termination analysis, the distinction between integers and rationals has already been considered, both regarding ranking-function generation [13, 22] and the very decidability of the termination problem [9, 16, 45]. All these works left the integer case open. Interestingly, our results provide an insight on how to make the solution proposed by Bradley et al. [13] complete (see Sec. 6).

This paper is organized as follows. Sec. 2 gives definitions and background information regarding linear constraint loops, linear ranking functions and the mathematical notions involved. Sec. 3 proves that the decision problem, denoted LINRF($\mathbb{Z}$), is coNP-complete, and also presents an exponential-time ranking-function synthesis algorithm. Sec. 4 discusses PTIME-solvable cases. Sec. 5 describes a prototype implementation. Sec. 6 surveys related previous work. Sec. 7 concludes.

2. Preliminaries

In this section we recall some results on (integer) polyhedra on which we will rely, define the kind of loops we are interested in, and formally define the *linear ranking function* problem.

2.1 Integer polyhedra

We recall some useful definitions and properties which can all be found in [41]. A *rational convex polyhedron* $\mathcal{P} \subseteq \mathbb{Q}^n$ (*polyhedron* for short) is the set of solutions of a set of inequalities $A\mathbf{x} \leq \mathbf{b}$, namely $\mathcal{P} = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{b}\}$, where $A \in \mathbb{Q}^{m\times n}$ is a rational matrix of n columns and m rows, $\mathbf{x} \in \mathbb{Q}^n$ and $\mathbf{b} \in \mathbb{Q}^m$ are column vectors of n and m rational values respectively. We say that $\mathcal{P}$ is specified by $A\mathbf{x} \leq \mathbf{b}$.

The set of *recession directions* of a polyhedron $\mathcal{P}$ specified by $A\mathbf{x} \leq \mathbf{b}$ is the set $\mathcal{R}_{\mathcal{P}} = \{\mathbf{y} \in \mathbb{Q}^n \mid A\mathbf{y} \leq \mathbf{0}\}$. For a given polyhedron $\mathcal{P} \subseteq \mathbb{Q}^n$ we let $I(\mathcal{P})$ be $\mathcal{P} \cap \mathbb{Z}^n$, i.e., the set of integer points of $\mathcal{P}$. The *integer hull* of $\mathcal{P}$, commonly denoted by $\mathcal{P}_I$, is defined as the convex hull of $I(\mathcal{P})$, i.e., every rational point of $\mathcal{P}_I$ is a convex combination of integer points. It is known that $\mathcal{P}_I$ is also a polyhedron. An *integer polyhedron* is a polyhedron $\mathcal{P}$ such that $\mathcal{P} = \mathcal{P}_I$. We also say that $\mathcal{P}$ is *integral*.

Polyhedra also have a *generator representation* in terms of vertices and rays, written as

$$\mathcal{P} = \operatorname{convhull}\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} + \operatorname{cone}\{\mathbf{y}_1, \ldots, \mathbf{y}_t\}\,.$$

This means that $\mathbf{x} \in \mathcal{P}$ iff $\mathbf{x} = \sum_{i=1}^{m} a_i\cdot\mathbf{x}_i + \sum_{j=1}^{t} b_j\cdot\mathbf{y}_j$ for some rationals $a_i, b_j \geq 0$, where $\sum_{i=1}^{m} a_i = 1$. Note that $\mathbf{y}_1, \ldots, \mathbf{y}_t$ are the recession directions of $\mathcal{P}$, i.e., $\mathbf{y} \in \mathcal{R}_{\mathcal{P}}$ iff $\mathbf{y} = \sum_{j=1}^{t} b_j \cdot \mathbf{y}_j$ for some rational $b_j \geq 0$. When $\mathcal{P}$ is integral, there is a generator representation in which all $\mathbf{x}_i$ and $\mathbf{y}_j$ are integer.

Complexity of algorithms on polyhedra is measured in this paper by running time, on a conventional computational model (polynomially equivalent to a Turing machine), as a function of the *bit-size* of the input. Following [41, Sec. 2.1], we define the bit-size of an integer x as $\|x\| = 1 + \lceil\log(|x| + 1)\rceil$; the bit-size of an n-dimensional vector $\mathbf{a}$ as $\|\mathbf{a}\| = n + \sum_{i=1}^{n} \|a_i\|$; and the bit-size of an inequality $\mathbf{a}\cdot\mathbf{x} \leq c$ as $1 + \|c\| + \|\mathbf{a}\|$.

For a polyhedron $\mathcal{P} \subseteq \mathbb{Q}^n$ defined by $A\mathbf{x} \leq \mathbf{b}$, we let $\|\mathcal{P}\|_b$ be the bit-size of $A\mathbf{x} \leq \mathbf{b}$, which we can take as the sum of the sizes of the inequalities. The *facet size*, denoted by $\|\mathcal{P}\|_f$, is the smallest number $\phi \geq n$ such that $\mathcal{P}$ may be described by *some* $A\mathbf{x} \leq \mathbf{b}$ where each inequality in $A\mathbf{x} \leq \mathbf{b}$ fits in ϕ bits. Clearly, $\|\mathcal{P}\|_f \leq \|\mathcal{P}\|_b$. The *vertex size*, denoted by $\|\mathcal{P}\|_v$, is the smallest number $\psi \geq n$ such that $\mathcal{P}$ has a generator representation in which each of $\mathbf{x}_i$ and $\mathbf{y}_j$ fits in ψ bits (the size of a vector is as above). For integer polyhedra, we restrict the generators to be integer. The following theorems may be found in [41, Th. 10.2, p. 121] and [41, Cor. 17.1a,17.1b, p. 238] (citing [32]) respectively.

THEOREM 2.1. *Let $\mathcal{P}$ be a rational polyhedron in $\mathbb{Q}^n$; then $\|\mathcal{P}\|_v \leq 4n^2\|\mathcal{P}\|_f$ and $\|\mathcal{P}\|_f \leq 4n^2\|\mathcal{P}\|_v$.*

THEOREM 2.2. *Let $\mathcal{P}$ be a rational polyhedron in $\mathbb{Q}^n$; then $\|\mathcal{P}_I\|_v \leq 6n^3\|\mathcal{P}\|_f$ and $\|\mathcal{P}_I\|_f \leq 24n^5\|\mathcal{P}\|_f$.*

2.2 Multipath linear-constraint loops

A *single-path* linear-constraint loop (SLC for short) over n variables $x_1, \ldots, x_n$ has the form

$$\textit{while } (B\mathbf{x} \leq \mathbf{b}) \textit{ do } A\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} \leq \mathbf{c} \tag{3}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ and $\mathbf{x}' = (x'_1, \ldots, x'_n)^{\mathrm{T}}$ are column vectors, and for some $p, q > 0$, $B \in \mathbb{Q}^{p\times n}$, $A \in \mathbb{Q}^{q\times 2n}$, $\mathbf{b} \in \mathbb{Q}^p$, $\mathbf{c} \in \mathbb{Q}^q$. The constraint $B\mathbf{x} \leq \mathbf{b}$ is called *the loop condition* (a.k.a. the loop guard) and the other constraint is called *the update*. The update is called *deterministic* if, for a given $\mathbf{x}$ (satisfying the loop condition) there is at most one $\mathbf{x}'$ satisfying the update constraint. The update is called *linear* if it can be rewritten as $\mathbf{x}' = A'\mathbf{x} + \mathbf{c}'$ for a matrix A' and vector $\mathbf{c}'$ of appropriate dimensions. We say that the loop is a *rational loop* if $\mathbf{x}$ and $\mathbf{x}'$ range over $\mathbb{Q}^n$, and that it is an *integer loop* if they range over $\mathbb{Z}^n$.

We say that there is a transition from a state $\mathbf{x} \in \mathbb{Q}^n$ to a state $\mathbf{x}' \in \mathbb{Q}^n$, if $\mathbf{x}$ satisfies the condition and $\mathbf{x}$ and $\mathbf{x}'$ satisfy the update. A transition can be seen as a point $\binom{\mathbf{x}}{\mathbf{x}'} \in \mathbb{Q}^{2n}$, where its first n components correspond to $\mathbf{x}$ and its last n components to $\mathbf{x}'$. For ease of notation, we denote such points by $\mathbf{x}'' = \binom{\mathbf{x}}{\mathbf{x}'}$. The set of all transitions $\mathbf{x}'' \in \mathbb{Q}^{2n}$ will be denoted, as a rule, by $\mathcal{Q}$. The transition polyhedron $\mathcal{Q}$ is specified by $A''\mathbf{x}'' \leq \mathbf{c}''$ where

$$A'' = \begin{pmatrix} B & 0 \\ & A \end{pmatrix} \qquad\qquad \mathbf{c}'' = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

Note that we may assume that $\mathcal{Q}$ does not include the origin, for if it includes it, the loop is clearly non-terminating (this condition is easy to check). Hence, $\mathcal{Q}$ is not a cone (i.e., $m \geq 1$ in the generator representation). The polyhedron defined by the loop condition $B\mathbf{x} \leq \mathbf{b}$ will be denoted by $\mathcal{C}$ (the condition polyhedron).

A *multipath* linear-constraint loop (MLC for short) differs by having alternative loop conditions and updates, which are, in principle, chosen non-deterministically (though the constraints may enforce a deterministic choice):

$$\mathit{loop}: \ \bigvee_{i=1}^{k} B_i\mathbf{x} \leq \mathbf{b}_i \Rightarrow A_i \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq \mathbf{c}_i$$

This means that the i-th update can be applied if the i-th condition is satisfied. Following the notation of SLC loops, the transitions of an MLC loop are specified by the transition polyhedra $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ is specified by $A_i''\mathbf{x}'' \leq \mathbf{c}_i''$. The polyhedron defined by the condition $B_i\mathbf{x} \leq \mathbf{b}_i$ is denoted by $\mathcal{C}_i$.

For simplifying the presentation, often we write loops with explicit equalities and inequalities instead of the matrix representation. We also might refer to loops by their corresponding transition polyhedra, or the sets of inequalities that define these polyhedra.

2.3 Linear ranking functions

A linear function $\rho : \mathbb{Q}^n \to \mathbb{Q}$ is of the form $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$ where $\vec{\lambda} \in \mathbb{Q}^n$ is a row vector and $\lambda_0 \in \mathbb{Q}$. For ease of notation we sometimes refer to a linear function using the row vector $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$. Next we define when a linear function is a *linear ranking function* (LRF for short) for a given rational or integer MLC loop.

DEFINITION 2.3. Given an MLC loop with $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ as transition polyhedra, and a linear function $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$. We say that ρ is a LRF for $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ iff the following hold for any rational point $\binom{\mathbf{x}}{\mathbf{x}'} \in \mathcal{Q}_i$

$$\vec{\lambda} \cdot \mathbf{x} + \lambda_0 \geq 0 \tag{4}$$

$$\vec{\lambda} \cdot (\mathbf{x} - \mathbf{x}') \geq 1 \tag{5}$$

and we say that it is a LRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$ iff (4,5) hold for any integer point $\binom{\mathbf{x}}{\mathbf{x}'} \in I(\mathcal{Q}_i)$.

Intuitively, (4) and (5) requires that $\rho(\mathbf{x}) \geq 0$ and $\rho(\mathbf{x}) - \rho(\mathbf{x}') \geq 1$ respectively. For *rational* loops this must hold for all rational transitions, and for *integer* loops it must hold for all integer transitions. Clearly, the existence of a LRF implies termination of the loop.

Note that in Def. 2.3 we require ρ to decrease at least by 1, where in the literature [38] this 1 is sometimes replaced by $\delta > 0$. It is easy to verify that these definitions are equivalent.

DEFINITION 2.4. The decision problem *Existence of a* LRF is defined by

Instance: an MLC loop.
Question: does there exist a LRF for this loop?

The decision problem is denoted by LINRF($\mathbb{Q}$) and LINRF($\mathbb{Z}$) for rational and integer loops respectively.

It is known that LINRF($\mathbb{Q}$) is PTIME-decidable [36, 38]. In this paper, we focus on LINRF($\mathbb{Z}$).

3. The general case is coNP-complete

In this section we show that the LINRF($\mathbb{Z}$) problem is coNP-complete; it is coNP-hard already for SLC loops. We also show that $LRFs$ can be synthesized in deterministic exponential time.

This section is organized as follows: in Sec. 3.1 we show that LINRF($\mathbb{Z}$) is coNP-hard; in Sec. 3.2 we show that it is in coNP for SLC loops, and in Sec. 3.3 for MLC loops; finally, in Sec. 3.4, we describe an algorithm for synthesizing $LRFs$.

3.1 coNP-hardness

We prove coNP-hardness in a strong form. A number problem (a problem whose instance is a matrix of integers) $\mathbf{Prob}$ is strongly hard for a complexity class, if there are polynomial reductions from all problems in that class to $\mathbf{Prob}$ such that the values of all numbers created by the reduction are polynomially bounded by the input bit-size. Assuming NP≠P, strongly NP-hard (or coNP-hard) problems cannot even have pseudo-polynomial algorithms [26].

THEOREM 3.1. *The* LINRF($\mathbb{Z}$) *problem is strongly coNP-hard, even for deterministic SLC loops.*

Proof. The problem of deciding whether a polyhedron given by $B\mathbf{x} \leq \mathbf{b}$ contains no integer point is a well-known coNP-hard problem (an easy reduction from SAT may be found in [31]). We reduce this problem to LINRF($\mathbb{Z}$). Given $B \in \mathbb{Z}^{m\times n}$ and $\mathbf{b} \in \mathbb{Z}^m$, we construct the following integer SLC loop

$$\mathit{while} \ \begin{pmatrix} B & -I \\ 0 & -I \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} \leq \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \ \mathit{do} \ \begin{pmatrix} \mathbf{x}' \\ \mathbf{z}' \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \end{pmatrix}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$, $\mathbf{z} = (z_1, \ldots, z_m)^{\mathrm{T}}$ are integer variables, and I is an identity matrix of size $m \times m$.

Suppose $B\mathbf{x} \leq \mathbf{b}$ has an integer solution $\mathbf{x}$. Then, the loop does not terminate when starting from this $\mathbf{x}$ and $\mathbf{z}$ set to $\mathbf{0}$, since the guard is satisfied and the update does not change the values. Thus, it does not have any ranking function, let along a LRF.

Next, suppose $B\mathbf{x} \leq \mathbf{b}$ does not have an integer solution. Then, for any initial state for which the loop guard is enabled it must hold that $z_1 + \cdots + z_m > 0$, for otherwise $z_1, \ldots, z_m$ must be 0 in which case the constraint $B\mathbf{x} - I\mathbf{z} \leq \mathbf{b}$ has no integer solution. Since the updated vector $\mathbf{z}'$ is deterministically set to $\mathbf{0}$, the guard will not be enabled in the next state, hence the loop terminates after one iteration. Clearly $z_1 + \cdots + z_m > z_1' + \cdots + z_m' = 0$, so we conclude that $z_1 + \cdots + z_m$ is a LRF. ☐

Note that in the above reduction we rely on the hardness of whether a given polyhedron is empty. This problem is coNP-hard even for bounded polyhedra (due to the reduction from SAT in which variables are bounded by 0 and 1). This means that even for loops that only manipulate integers in a rather small range, the problem is coNP-hard. The parameter "responsible" for the exponential behavior in this case is the number of variables.

3.2 Inclusion in coNP for SLC loops

To prove that LINRF($\mathbb{Z}$) is in coNP, we show that the complement of LINRF($\mathbb{Z}$), the problem of *nonexistence* of a LRF, is in NP, that is, has a polynomially-checkable witness. In what follows we assume as input an SLC loop with a transition polyhedron $\mathcal{Q} \subseteq \mathbb{Q}^{2n}$. The input is given as the set of linear inequalities $A''\mathbf{x}'' \leq \mathbf{c}''$ that define $\mathcal{Q}$. The proof follows the following lines:

1. We show that there is no LRF for $I(\mathcal{Q})$ iff there is a *witness* that consists of two sets of integer points $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$, such that a certain set of inequalities $\Psi_{WS}(X,Y)$ has no solution over the rationals; and
2. We show that if there is a witness then there is one with bit-size polynomial in the input bit-size.

To make sense of the following definitions, think of a vector $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ as a "candidate LRF" that we may want to verify (or, in our case, to eliminate).

DEFINITION 3.2. We say that $\mathbf{x}'' = \left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{x}'\end{smallmatrix}\right) \in I(\mathcal{Q})$ is a witness *against* $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ if it fails to satisfy at least one of conditions (4) and (5). The set of $(\lambda_0,\vec{\lambda})$ witnessed against by $\mathbf{x}''$ is denoted by $W(\mathbf{x}'')$.

DEFINITION 3.3. We say that $\mathbf{y}'' = \left(\begin{smallmatrix}\mathbf{y}\\ \mathbf{y}'\end{smallmatrix}\right) \in I(\mathcal{R}_{\mathcal{Q}})$ is a *homogeneous component of a witness* (h-witness) against $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ if it fails to satisfy at least one of

$$\vec{\lambda}\cdot\mathbf{y} \geq 0 \tag{6}$$

$$\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}') \geq 0 \tag{7}$$

The set of $(\lambda_0,\vec{\lambda})$ h-witnessed against by $\mathbf{y}''$ is denoted by $W_H(\mathbf{y}'')$.

The meaning of the witness of Def. 3.2 is quite straightforward. Let us intuitively explain the meaning of an h-witness. Suppose that $\mathbf{x}''$ is a point in $\mathcal{Q}_I$, and $\mathbf{y}''$ is a ray of $\mathcal{Q}_I$. Then a LRF ρ has to satisfy (4) for any integer point of the form $\mathbf{x}''+a\cdot\mathbf{y}''$ with $a>0$; letting a grow to infinity, we see that (4) implies the homogeneous inequality (6). Similarly, (5) implies (7).

DEFINITION 3.4. The set of $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ witnessed and h-witnessed against respectively by $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$ is defined as

$$WS(X,Y) = \bigcup_{\mathbf{x}''\in X} W(\mathbf{x}'') \;\cup\; \bigcup_{\mathbf{y}''\in Y} W_H(\mathbf{y}''). \tag{8}$$

LEMMA 3.5. *Let $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$. If $WS(X,Y) = \mathbb{Q}^{n+1}$, then there is no LRF for $I(\mathcal{Q})$.*

Proof. Let $WS(X,Y) = \mathbb{Q}^{n+1}$. For any $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$, we prove that $\rho(\mathbf{x}) = \vec{\lambda}\cdot\mathbf{x}+\lambda_0$ is not a LRF. If $(\lambda_0,\vec{\lambda}) \in W(\mathbf{x}'')$ for some $\mathbf{x}'' \in X$, then the conclusion is clear since conditions (4,5) do not hold. Otherwise, suppose that $(\lambda_0,\vec{\lambda}) \in W_H(\mathbf{y}'')$ for $\mathbf{y}'' \in Y$. Thus, $\mathbf{y}''$ fails to satisfy one of conditions (6,7). Next we show that, in such case, there must exists $\mathbf{z}'' \in I(\mathcal{Q})$ that fails either (4) or (5).

Case 1: Suppose (6) is not satisfied. That is, $\vec{\lambda}\cdot\mathbf{y} < 0$.

Choose an arbitrary $\mathbf{x}'' = \left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{x}'\end{smallmatrix}\right) \in I(\mathcal{Q})$, and assume that $\rho(\mathbf{x}) \geq 0$, since otherwise ρ fails (4) and is not a LRF. Note that for any integer $a \geq 0$, the integer point $\mathbf{z}'' = \mathbf{x}''+a\cdot\mathbf{y}''$ is a transition in $I(\mathcal{Q})$, and $\mathbf{z}'' = \left(\begin{smallmatrix}\mathbf{z}\\ \mathbf{z}'\end{smallmatrix}\right) = \left(\begin{smallmatrix}\mathbf{x}+a\cdot\mathbf{y}\\ \mathbf{x}'+a\cdot\mathbf{y}'\end{smallmatrix}\right)$. We choose a as an integer sufficiently large so that $a\cdot(\vec{\lambda}\cdot\mathbf{y}) \leq -(1+\rho(\mathbf{x}))$. Now,

$$\begin{aligned}\rho(\mathbf{z}) &= \vec{\lambda}\cdot(\mathbf{x}+a\cdot\mathbf{y})+\lambda_0\\ &= \rho(\mathbf{x})+a\cdot(\vec{\lambda}\cdot\mathbf{y}) \leq \rho(\mathbf{x})-(1+\rho(\mathbf{x})) = -1\end{aligned}$$

So ρ fails (4) on $\mathbf{z}'' \in I(\mathcal{Q})$, and thus cannot be a LRF.

Case 2: Suppose (7) is not satisfied. That is, $\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}') < 0$.

Choose an arbitrary $\mathbf{x}'' = \left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{x}'\end{smallmatrix}\right) \in I(\mathcal{Q})$, and assume that $\rho(\mathbf{x})-\rho(\mathbf{x}') \geq 1$, since otherwise ρ fails (5) and is not a LRF. Define $\mathbf{z}''$ as above, but now choosing a sufficiently large to make $a\cdot(\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}')) \leq -(1+\rho(\mathbf{x})-\rho(\mathbf{x}'))$. Now,

$$\begin{aligned}\rho(\mathbf{z})-\rho(\mathbf{z}') &= \vec{\lambda}\cdot((\mathbf{x}+a\cdot\mathbf{y})-(\mathbf{x}'+a\cdot\mathbf{y}'))\\ &= \rho(\mathbf{x})-\rho(\mathbf{x}')+a\cdot(\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}'))\\ &\leq \rho(\mathbf{x})-\rho(\mathbf{x}')-(1+\rho(\mathbf{x})-\rho(\mathbf{x}')) = -1\end{aligned}$$

So ρ fails (5) on $\mathbf{z}'' \in I(\mathcal{Q})$, and thus cannot be a LRF. □

Note that the condition $WS(X,Y) = \mathbb{Q}^{n+1}$ is equivalent to saying that the conjunction of inequalities (4,5), for all $\mathbf{x}'' \in X$, and inequalities (6,7), for all $\mathbf{y}'' \in Y$, has no (rational) solution. We denote this set of inequalities by $\Psi_{WS}(X,Y)$. Note that the variables in $\Psi_{WS}(X,Y)$ are $\lambda_0,\ldots,\lambda_n$, which range over $\mathbb{Q}$, and thus, the test that it has no solution can be done in polynomial time since it is an LP problem over the rationals.

EXAMPLE 3.6. Consider the following integer SLC loop:

$$\textit{while } (x_1 \geq 0) \textit{ do } x_1' = x_1+x_2, x_2' = x_2-1$$

Let $\mathbf{x}_1''=(0,2,2,1)^{\mathrm{T}} \in I(\mathcal{Q})$ and $\mathbf{y}_1''=(1,-2,-1,-2)^{\mathrm{T}} \in I(\mathcal{R}_{\mathcal{Q}})$. Then, $\Psi_{WS}(\{\mathbf{x}_1''\},\{\mathbf{y}_1''\})$ is a conjunction of the inequalities

$$\{2\lambda_2+\lambda_0 \geq 0,\ -2\lambda_1+\lambda_2 \geq 1,\ \lambda_1-2\lambda_2 \geq 0,\ 2\lambda_1 \geq 0\} \tag{9}$$

The first two inequalities correspond to applying (4,5) to $\mathbf{x}_1''$, and the other ones to applying (6,7) to $\mathbf{y}_1''$. It is easy to verify that (9) is not satisfiable, thus, $WS(\{\mathbf{x}_1''\},\{\mathbf{y}_1''\}) = \mathbb{Q}^3$ and the loop does not have a LRF. This is a classical loop for which there is no LRF.

Lemma 3.5 provides a sufficient condition for the nonexistence of LRF, the next lemma shows that this condition is necessary. In particular, it shows that if there is no LRF for $I(\mathcal{Q})$, then the vertices and rays of $\mathcal{Q}_I$ serve as X and Y of Lemma 3.5.

LEMMA 3.7. *Let the integer hull of the transition polyhedron $\mathcal{Q}$ be $\mathcal{Q}_I$=convhull$\{\mathbf{x}_1'',\ldots,\mathbf{x}_m''\}$+cone$\{\mathbf{y}_1'',\ldots,\mathbf{y}_t''\}$. If there is no LRF for $I(\mathcal{Q})$, then $WS(\{\mathbf{x}_1'',\ldots,\mathbf{x}_m''\},\{\mathbf{y}_1'',\ldots,\mathbf{y}_t''\})=\mathbb{Q}^{n+1}$.*

Proof. We prove the contra-positive. Suppose that

$$WS(\{\mathbf{x}_1'',\ldots,\mathbf{x}_m''\},\{\mathbf{y}_1'',\ldots,\mathbf{y}_t''\}) \neq \mathbb{Q}^{n+1}.$$

Then, there is $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ that fulfills (4,5) for all $\mathbf{x}_i''$ and (6,7) for all $\mathbf{y}_j''$. We claim that $\rho(\mathbf{x}) = \vec{\lambda}\cdot\mathbf{x}+\lambda_0$ is a LRF for $I(\mathcal{Q})$.

To see this, let $\mathbf{x}'' = \left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{x}'\end{smallmatrix}\right)$ be an arbitrary point of $I(\mathcal{Q})$. Then $\mathbf{x}'' = \sum_{i=1}^m a_i\cdot\mathbf{x}_i''+\sum_{j=1}^t b_j\cdot\mathbf{y}_j''$ for some $a_i,b_j \geq 0$ where $\sum_{i=1}^m a_i = 1$. Now, we show that $\mathbf{x}''$ and ρ satisfy (4,5) which means that ρ is a LRF for $I(\mathcal{Q})$:

$$\begin{aligned}\vec{\lambda}\cdot\mathbf{x}+\lambda_0 &= \lambda_0+\sum_{i=1}^m a_i\cdot(\vec{\lambda}\cdot\mathbf{x}_i)+\sum_{j=1}^t b_j\cdot(\vec{\lambda}\cdot\mathbf{y}_j)\\ &= \sum_{i=1}^m a_i\cdot(\vec{\lambda}\cdot\mathbf{x}_i+\lambda_0)+\sum_{j=1}^t b_j\cdot(\vec{\lambda}\cdot\mathbf{y}_j)\\ &\geq 0+0 = 0\\ \vec{\lambda}\cdot(\mathbf{x}-\mathbf{x}') &= \sum_{i=1}^m a_i\cdot(\vec{\lambda}\cdot(\mathbf{x}_i-\mathbf{x}_i'))+\sum_{j=1}^t b_j\cdot(\vec{\lambda}\cdot(\mathbf{y}_j-\mathbf{y}_j'))\\ &\geq 1+0 = 1\end{aligned}$$

□

Note that the solutions of $\Psi_{WS}(\{\mathbf{x}_1'',\ldots,\mathbf{x}_m''\},\{\mathbf{y}_1'',\ldots,\mathbf{y}_t''\})$ in Lemma 3.7 define the set of all $LRFs$ for $I(\mathcal{Q})$ (see Sec. 3.4).

EXAMPLE 3.8. Consider again the loop of Ex. 3.6, and recall that it does not have a LRF. The generator representation of $\mathcal{Q}_I$ is

$$\mathcal{Q}_I = \text{convhull}\{\mathbf{x}_1''\}+\text{cone}\{\mathbf{y}_1'',\mathbf{y}_2'',\mathbf{y}_3''\}$$

where $\mathbf{x}_1''=(0,1,1,0)^{\mathrm{T}}$, $\mathbf{y}_1''=(0,-1,-1,-1)^{\mathrm{T}}$, $\mathbf{y}_2''=(0,1,1,1)^{\mathrm{T}}$ and $\mathbf{y}_3''=(1,-1,0,-1)^{\mathrm{T}}$. Then, $\Psi_{WS}(\{\mathbf{x}_1''\},\{\mathbf{y}_1'',\mathbf{y}_2'',\mathbf{y}_3''\})$ is a conjunction of the following inequalities

$$\begin{array}{cccc}\lambda_2+\lambda_0 \geq 0 & -\lambda_2 \geq 0 & \lambda_2 \geq 0 & \lambda_1-\lambda_2 \geq 0\\ -\lambda_1+\lambda_2 \geq 1 & \lambda_1 \geq 0 & -\lambda_1 \geq 0 & \lambda_1 \geq 0\end{array} \tag{10}$$

The inequalities in the leftmost column correspond to applying (4,5) to $\mathbf{x}_1''$, and those in the other columns to applying (6,7) to $\mathbf{y}_1''$, $\mathbf{y}_2''$, and $\mathbf{y}_3''$ respectively. It is easy to verify that (10) is not satisfiable. Thus, $WS(\{\mathbf{x}_1''\},\{\mathbf{y}_1'',\mathbf{y}_2'',\mathbf{y}_3''\}) = \mathbb{Q}^3$.

COROLLARY 3.9. *There is no LRF for $I(\mathcal{Q})$, iff there are two sets $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{P}})$ such that $WS(X,Y) = \mathbb{Q}^{n+1}$.*

The next lemma concerns the bit-size of the witness.

LEMMA 3.10. *If there exists a witness for the nonexistence of LRF for $I(\mathcal{Q})$, there exists one with $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$ such that $|X \cup Y| \leq n+1$; and the bit-size of $X \cup Y$ is polynomial in the bit-size of the input.*

Proof. Recall that by Lemma 3.7, if $I(\mathcal{Q})$ has no LRF, then

$$WS(\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\}) = \mathbb{Q}^{n+1}$$

or, equivalently, $\Psi_{WS}(\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\})$ has no solution. A corollary of Farkas' Lemma [41, p. 94] states that: if a set of inequalities over $\mathbb{Q}^n$ has no solution, there is a subset of at most n inequalities that has no solution. Since the set of inequalities $\Psi_{WS}(\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\})$ is over $\mathbb{Q}^{n+1}$, there is a subset of at most $n+1$ inequalities that has no solution. This subset involves at most $n+1$ integer points out of $\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\}$ and $\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\}$, because every inequality in $\Psi_{WS}(\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\})$ is defined by either one $\mathbf{x}''_i$ or $\mathbf{y}''_i$ (see eqs. (4–7)). Let these points be $X \cup Y$, then $|X \cup Y| \leq n+1$ and $\Psi_{WS}(X,Y)$ has no solution, i.e., $WS(X,Y) = \mathbb{Q}^{n+1}$.

Now we show that $X \cup Y$ may be chosen to have bit-size polynomial in the size of the input. Recall that the input is the set of inequalities $A''\mathbf{x}'' \leq \mathbf{b}$ that define $\mathcal{Q}$, and its bit-size is $\|\mathcal{Q}\|_b$. Recall that the points of $X \cup Y$ in Lemma 3.7 come from the generator representation, and that there is a generator representation in which each vertex/ray can fit in $\|\mathcal{Q}_I\|_v$ bits. Thus, the bit-size of $X \cup Y$ may be bounded by $(n+1) \cdot \|\mathcal{Q}_I\|_v$. By Th. 2.2, since the dimension of $\mathcal{Q}$ is $2n$,

$$(n+1) \cdot \|\mathcal{Q}_I\|_v \leq (n+1) \cdot (6 \cdot (2n)^3 \cdot \|\mathcal{Q}\|_f) \leq 96n^4 \cdot \|\mathcal{Q}\|_b$$

which is polynomial in the bit-size of the input. □

EXAMPLE 3.11. Consider $\Psi_{WS}(\{\mathbf{x}''_1\},\{\mathbf{y}''_1, \mathbf{y}''_2, \mathbf{y}''_3\})$ of Ex. 3.8. It is easy to see that the inequalities $-\lambda_2 \geq 0$, $\lambda_1 \geq 0$ and $-\lambda_1 + \lambda_2 \geq 1$ are enough for unsatisfiability ($n+1$ inequalities, since $n = 2$). These inequalities correspond to $\mathbf{x}''_1$ and $\mathbf{y}''_1$, and thus, they form a witness for the nonexistence of LRF.

THEOREM 3.12. $\text{LinRF}(\mathbb{Z}) \in \text{coNP}$ *for SLC loops.*

Proof. We show that the complement of $\text{LinRF}(\mathbb{Z})$ has a polynomially checkable witness. The witness is a listing of sets X and Y of at most $n+1$ elements and has a polynomial bit-size (specifically, a bit-size bounded as in Lemma 3.10). Verifying a witness consists of the following steps:

Step 1: Verify that each $\mathbf{x}'' \in X$ is in $I(\mathcal{Q})$, which can be done by verifying $A''\mathbf{x}'' \leq \mathbf{c}''$; and that each $\mathbf{y}'' \in Y$ is in $I(\mathcal{R}_{\mathcal{Q}})$, which can be done by verifying $A''\mathbf{y}'' \leq 0$. This is done in polynomial-time. Note that according to Lemma 3.5 it is not necessary to check that X and Y come from a particular generator representation.

Step 2: Verify that $WS(X,Y) = \mathbb{Q}^{n+1}$. This can be done by checking that $\Psi_{WS}(X,Y)$ has no solutions, which can be done in polynomial-time since it is an LP problem over $\mathbb{Q}^{n+1}$. □

3.3 Inclusion in coNP for MLC loops

In this section we consider the inclusion in coNP for MLC loops. For this, we assume an input MLC loop with transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$ where each $\mathcal{Q}_i$ is specified by $A''_i\mathbf{x}'' \leq \mathbf{c}''_i$.

The proof follows the structure of the SLC case. The main difference is that points of the witness $X \cup Y$ may come from different transition polyhedra. Namely, $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$ where each $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$. We rewrite Lemmas 3.5, 3.7, and 3.10, Cor. 3.9, and Th. 3.12 in terms of such witnesses (the proofs are the same unless stated otherwise).

LEMMA 3.13. *Let $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$, where $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$. If $WS(X,Y) = \mathbb{Q}^{n+1}$, then there is no LRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$.*

LEMMA 3.14. *For $1 \leq i \leq k$, let $\mathcal{Q}_{iI} = \text{convhull}\{X_i\} + \text{cone}\{Y_i\}$ be the integer hull of $\mathcal{Q}_i$, and define $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$. If there is no LRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, then $WS(X,Y) = \mathbb{Q}^{n+1}$.*

Proof. The proof follows that of Lemma 3.7. We pick $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1} \setminus WS(X,Y)$ and show that $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$ is a LRF for each $I(\mathcal{Q}_i)$. This is accomplished by performing the same calculation, however referring to X_i and Y_i when proving that ρ is a LRF for $I(\mathcal{Q}_i)$. □

COROLLARY 3.15. *There is no LRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, iff there are two sets $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$, where $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$, such that $WS(X,Y) = \mathbb{Q}^{n+1}$.*

LEMMA 3.16. *If there exists a witness for the nonexistence of LRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, then there exists one, with $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$, where $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$, such that $|X \cup Y| \leq n+1$; and the bit-size of $X \cup Y$ is polynomial in the bit-size of the input.*

THEOREM 3.17. $\text{LinRF}(\mathbb{Z}) \in \text{coNP}$.

Proof. The difference from that of Th. 3.12 is that the witness is given as $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$. Thus, the verifier should use the appropriate set of inequalities to check that each $\mathbf{x}'' \in X_i$ is in $I(\mathcal{Q}_i)$, and that each $\mathbf{y}'' \in Y_i$ is in $I(\mathcal{R}_{\mathcal{Q}_i})$. □

EXAMPLE 3.18. Consider the following integer MLC loop

$$\begin{array}{lllll} \textit{loop}: & & x_1 \geq 0, x_2 \geq 0 & \Rightarrow & x'_1 = x_1 - 1 \\ & \vee & x_1 \geq 0, x_2 \geq 0 & \Rightarrow & x'_2 = x_2 - 1, x'_1 = x_1 \end{array}$$

It is a classical MLC loop for which there is no LRF. The integer hull of the corresponding transition polyhedra are

$$\begin{array}{ll} \mathcal{Q}_{1I} = & \text{convhull}\{\mathbf{x}''_1\} + \text{cone}\{\mathbf{y}''_1, \mathbf{y}''_2, \mathbf{y}''_3, \mathbf{y}''_4\} \\ \mathcal{Q}_{2I} = & \text{convhull}\{\mathbf{x}''_2\} + \text{cone}\{\mathbf{y}''_5, \mathbf{y}''_6\} \end{array}$$

where $\mathbf{x}''_1 = (0,0,-1,0)^{\mathrm{T}}$, $\mathbf{x}''_2 = (0,0,0,-1)^{\mathrm{T}}$, $\mathbf{y}''_1 = (0,0,0,-1)^{\mathrm{T}}$, $\mathbf{y}''_2 = (0,0,0,1)^{\mathrm{T}}$, $\mathbf{y}''_3 = (0,1,0,0)^{\mathrm{T}}$, $\mathbf{y}''_4 = (1,0,1,0)^{\mathrm{T}}$, $\mathbf{y}''_5 = (0,1,0,1)^{\mathrm{T}}$ and $\mathbf{y}''_6 = (1,0,1,0)^{\mathrm{T}}$. Let us first consider each path separately. We get

$$\Psi_{WS}(\{\mathbf{x}''_1\},\{\mathbf{y}''_1, \mathbf{y}''_2, \mathbf{y}''_3\}) = \left\{ \begin{array}{l} \lambda_0 \geq 0, \lambda_1 \geq 1, \\ \lambda_2 \geq 0, -\lambda_2 \geq 0 \end{array} \right\} \quad (11)$$

$$\Psi_{WS}(\{\mathbf{x}''_2\},\{\mathbf{y}''_4, \mathbf{y}''_5, \mathbf{y}''_6\}) = \{\lambda_0 \geq 0, \lambda_2 \geq 1\} \quad (12)$$

Both (11) and (12) are satisfiable. In fact, their solutions define the corresponding $LRFs$ for each path when considered separately. For the MLC loop, we have that $\Psi_{WS}(\{\mathbf{x}''_1, \mathbf{x}''_2\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_6\})$ is the conjunction of the inequalities in (11) and (12), which is not satisfiable. Thus, while each path has a LRF, the MLC loop does not. Note that the inequalities $\lambda_2 \geq 1$ and $-\lambda_2 \geq 0$ are enough to get unsatisfiability of (11,12), thus, $X = \{\mathbf{x}''_2\}$ and $Y = \{\mathbf{y}''_2\}$ is a witness, and consists of less than $n+1$ points ($n = 2$ in this case).

3.4 Synthesizing a linear ranking function

Although the existence of a LRF suffices for proving termination, generating a complete representation of the LRF is important in some contexts, for instance complexity analysis where a LRF provides an upper bound on the number of iterations that a loop can perform. In this section we give a complete algorithm that generates $LRFs$ for MLC loops given by transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$. The following lemma is directly implied by lemmas 3.13 and 3.14.

LEMMA 3.19. *For $1 \leq i \leq k$, let the integer hull of $\mathcal{Q}_i$ be $\mathcal{Q}_{iI} = \text{convhull}\{X_i\} + \text{cone}\{Y_i\}$, and define $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$. Then, $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$ is a LRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, iff $(\lambda_0, \vec{\lambda})$ is a solution for $\Psi_{WS}(X,Y)$.*

The following algorithm is clearly implied: (1) Compute the generator representation for each $\mathcal{Q}_{iI}$; (2) Construct $\Psi_{WS}(X,Y)$; and (3) Use LP to find a solution $(\lambda_0, \vec{\lambda})$ for $\Psi_{WS}(X,Y)$.

EXAMPLE 3.20. Consider again Loop (1) from Sec. 1. The integer hull of the transition polyhedron is

$$\mathcal{Q}_I = \text{convhull}\{\mathbf{x}_1'', \mathbf{x}_2''\} + \text{cone}\{\mathbf{y}_1'', \mathbf{y}_2''\}$$

where $\mathbf{x}_1'' = (1,1,1,0)^{\mathsf{T}}$, $\mathbf{x}_2'' = (1,0,1,-1)^{\mathsf{T}}$, $\mathbf{y}_1''=(1,1,1,-1)^{\mathsf{T}}$, and $\mathbf{y}_2'' = (1,-1,1,-3)^{\mathsf{T}}$. Then, $\Psi_{WS}(\{\mathbf{x}_1'', \mathbf{x}_2''\},\{\mathbf{y}_1'', \mathbf{y}_2''\})$ is the conjunction of the following inequalities (we eliminated clearly redundant inequalities)

$$\begin{Bmatrix} \lambda_1 + \lambda_2 + \lambda_0 \geq 0, & \lambda_1 + \lambda_0 \geq 0, & \\ \lambda_1 + \lambda_2 \geq 0, & \lambda_1 - \lambda_2 \geq 0, & \lambda_2 \geq 1 \end{Bmatrix} \tag{13}$$

which is satisfiable for $\lambda_1 = \lambda_2 = 1$ and $\lambda_0 = -1$, and therefore, $f(x_1, x_2) = x_1 + x_2 - 1$ is a LRF. Recall that this loop does not terminate when the variables range over $\mathbb{Q}$, e.g., for $x_1 = \frac{1}{2}$ and $x_2 = \frac{1}{2}$ (see Fig. 1(A)).

Let us consider now Loop (2) from Sec. 1. The integer hull of the transition polyhedron is

$$\mathcal{Q}_I = \text{convhull}\{\mathbf{x}_1'', \mathbf{x}_2'', \mathbf{x}_3'', \mathbf{x}_4'', \mathbf{x}_5'', \mathbf{x}_6''\} + \text{cone}\{\mathbf{y}_1'', \mathbf{y}_2''\}$$

where $\mathbf{x}_1'' = (4,16,1,16)^{\mathsf{T}}$, $\mathbf{x}_2'' = (1,4,0,4)^{\mathsf{T}}$, $\mathbf{x}_3'' = (2,8,1,8)^{\mathsf{T}}$, $\mathbf{x}_4'' = (1,1,0,1)^{\mathsf{T}}$, $\mathbf{x}_5'' = (4,1,1,1)^{\mathsf{T}}$, $\mathbf{x}_6'' = (2,1,1,1)^{\mathsf{T}}$, $\mathbf{y}_1'' = (5,0,2,0)^{\mathsf{T}}$ and $\mathbf{y}_2 = (5,20,2,20)^{\mathsf{T}}$. Then, the set of inequalities $\Psi_{WS}(\{\mathbf{x}_1'', \ldots, \mathbf{x}_6''\},\{\mathbf{y}_1'', \mathbf{y}_2''\})$ is the conjunction of the following inequalities (we eliminated clearly redundant inequalities)

$$\begin{Bmatrix} \lambda_1 \geq 1, \ 4\lambda_1+\lambda_2+\lambda_0 \geq 0, \ 4\lambda_1+16\lambda_2+\lambda_0 \geq 0, \\ 2\lambda_1+\lambda_2+\lambda_0 \geq 0, \ \lambda_1+4\lambda_2+\lambda_0 \geq 0, \\ 2\lambda_1+8\lambda_2+\lambda_0 \geq 0, \ 5\lambda_1+20\lambda_2 \geq 0, \ \lambda_1+\lambda_2+\lambda_0 \geq 0 \end{Bmatrix} \tag{14}$$

which is satisfiable for $\lambda_1 = 1$, $\lambda_2 = 0$ and $\lambda_0 = -1$, and therefore, $f(x_1, x_2) = x_1 - 1$ is a LRF. Recall that this loop, too, does not terminate when the variables range over $\mathbb{Q}$, e.g., for $x_1 = \frac{1}{4}$ and $x_2 = 1$ (see Fig. 1(C)).

If we consider both loops (1) and (2) as two paths in an MLC loop, then to synthesize $LRFs$ we use the conjunction of the inequalities in (13) and (14). In this case, $\lambda_1 = \lambda_2 = 1$ and $\lambda_0 = -1$, is a solution, but $\lambda_1 = 1$, $\lambda_2 = 0$ and $\lambda_0 = -1$ is not. Therefore, $f(x_1, x_2) = x_1 + x_2 - 1$ is a LRF for both paths, and thus for the MLC loop, but not $f(x_1, x_2) = x_1 - 1$.

Given our hardness results, one cannot expect a polynomial-time algorithm. Indeed, constructing the generator representation of the integer hull of a polyhedron from the corresponding set of inequalities $A_i''\mathbf{x} \leq \mathbf{c}_i''$ may require exponential time—the number of generators itself may be exponential. Their bit-size, on the other hand, is polynomial by Th. 2.2. This is interesting, since it yields:

COROLLARY 3.21. *Given an MLC loop with transition polyhedra $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ is specified by $A_i''\mathbf{x} \leq \mathbf{c}_i''$. If there is a LRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, there is one whose bit-size is polynomial in the bit-size of $A_i''\mathbf{x} \leq \mathbf{c}_i''$, namely in $\max_i \|\mathcal{Q}_i\|_b$.*

Proof. As in the last section, we bound the bit-size of each of the generators of $\mathcal{Q}_{iI}$ by $\|\mathcal{Q}_{iI}\|_v \leq 6(2n)^3 \cdot \|\mathcal{Q}_i\|_f \leq 48n^3 \cdot \|\mathcal{Q}_i\|_b$ for an appropriate i. This means that the bit-size of each equation in $\Psi_{WS}(X,Y)$, having one of the forms (4), (5), (6), or (7) is at most $5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b)$. Let $\mathcal{P}$ be the polyhedron defined by $\Psi_{WS}(X,Y)$, then $\|\mathcal{P}\|_f \leq 5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b)$. If $\Psi_{WS}(X,Y)$ has a solution, then any vertex of $\mathcal{P}$ is such a solution, and yields a LRF. Using Th. 2.1, together with the above bound for $\|\mathcal{P}\|_f$ and the fact that the dimension of $\mathcal{P}$ is $n+1$, we conclude that there is a generator representation for $\mathcal{P}$ in which the bit-size $\|\mathcal{P}\|_v$ of the vertices is bounded as follows:

$$\|\mathcal{P}\|_v \leq 4 \cdot (n+1)^2 \cdot \|\mathcal{P}\|_f \leq 4 \cdot (n+1)^2 \cdot (5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b))$$

This also bounds the bit-size of the corresponding LRF. □

We conclude this section by noting that the algorithm induced by Lemma 3.19 works also for LINRF($\mathbb{Q}$), if we consider $\mathcal{Q}_i$ instead of $\mathcal{Q}_{iI}$. This can be easily proven by reworking the proofs of Lemmas 3.13 and 3.14 for the case of $\mathcal{Q}_i$ instead of $\mathcal{Q}_{iI}$. We did not develop this line since the main use of these lemmas is proving the coNP-completeness for LINRF($\mathbb{Z}$). This, however, has an interesting consequence: LINRF($\mathbb{Q}$) is still PTIME even if the input loop is given in the generator representations form instead of the constraints form. Practically, implementations of polyhedra that use of the *double description method*, such as the Parma Polyhedra Library [5], in which both the generators and constraint representations are kept at the same time, can use the algorithm of Lemma 3.19 judiciously when it seems better than algorithms that use the constraints representation [36, 38].

4. Special cases in PTIME

In this section we discuss cases in which the LINRF($\mathbb{Z}$) problem is PTIME-decidable. We start by a basic observation: when the transition polyhedron of an SLC loop is *integral*, the LINRF($\mathbb{Z}$) and LINRF($\mathbb{Q}$) problems are equivalent (a very similar statement appears in [22, Lemma 3]).

LEMMA 4.1. *Let $\mathcal{Q}$ be a transition polyhedron of a given SLC loop, and let ρ be a linear function. If $\mathcal{Q}$ is integral, then ρ is a LRF for $\mathcal{Q}$ iff ρ is a LRF for $I(\mathcal{Q})$.*

Proof. Let $\mathcal{Q}$ be integral. ($\Rightarrow$) Suppose that ρ is a LRF for $\mathcal{Q}$, then it is also a LRF for $I(\mathcal{Q})$ since $I(\mathcal{Q}) \subseteq \mathcal{Q}$. ($\Leftarrow$) Suppose that ρ is a LRF for $I(\mathcal{Q})$, it thus satisfies (4,5) of Def. 2.3 for any integer point in $\mathcal{Q}$. However, by definition of an integer polyhedron, every rational point in $\mathcal{Q}$ is a convex combination of integer points from $I(\mathcal{Q})$, this proves that ρ satisfies conditions (4,5) for any rational point as well. To see this, choose an arbitrary rational point $\mathbf{x}'' = \left(\begin{smallmatrix} \mathbf{x} \\ \mathbf{x}' \end{smallmatrix}\right) \in \mathcal{Q}$. It can be written as $\mathbf{x}'' = \sum a_i \cdot \mathbf{x}_i''$ where $a_i \geq 0$, $\sum a_i = 1$ and $\mathbf{x}_i'' \in I(\mathcal{Q})$. Thus, $\mathbf{x}'' = \left(\begin{smallmatrix} \mathbf{x} \\ \mathbf{x}' \end{smallmatrix}\right) = \left(\begin{smallmatrix} \sum a_i \cdot \mathbf{x}_i \\ \sum a_i \cdot \mathbf{x}_i' \end{smallmatrix}\right)$, and

$$\begin{aligned} \rho(\mathbf{x}) &= (\vec{\lambda} \cdot \sum a_i \cdot \mathbf{x}_i) + \lambda_0 \\ &= \sum a_i \cdot (\vec{\lambda} \cdot \mathbf{x}_i + \lambda_0) \geq 0 \\ \rho(\mathbf{x}) - \rho(\mathbf{x}') &= (\vec{\lambda} \cdot \sum a_i \cdot \mathbf{x}_i) - (\vec{\lambda} \cdot \sum a_i \cdot \mathbf{x}_i') \\ &= \sum a_i \cdot \vec{\lambda} \cdot (\mathbf{x}_i - \mathbf{x}_i') \geq 1 \end{aligned}$$

□

The above lemma provides an alternative *complete* procedure for LINRF($\mathbb{Z}$), namely, compute a constraint representation of its integer hull Q_I and solve LINRF($\mathbb{Q}$). Note that computing the integer hull might require exponential time, and might also result in a polyhedron with an exponentially big description. This means that the above procedure is exponential in general; but this concern is circumvented if the transition polyhedron is integral to begin with; and in special cases where it is known that computing the integer hull is easy. Formally, we call a class of polyhedra *easy* if computing its integer hull can be done in polynomial time.

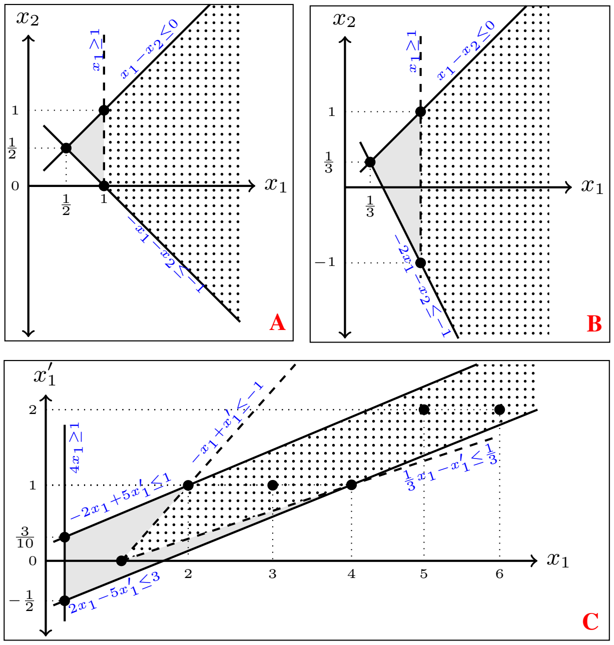

Figure 1. The polyhedra associated with three of our examples, projected to two dimensions. Dashed lines are added when computing the integer hull; dotted areas represent the integer hull; gray areas are rational points eliminated when computing the integer hull.

EXAMPLE 4.2. Consider again Loop (2) of Sec. 1. The transition polyhedron is not integral, computing its integer hull adds the inequalities $-x_1 + x_1' \leq -1$ and $\frac{1}{3}x_1 - x_1' \leq \frac{1}{3}$ (see Fig. 1(C)). Applying LINRF($\mathbb{Q}$) on this loop does not find a LRF since it does not terminate when the variables range over $\mathbb{Q}$, however, applying it on the integer hull finds the LRF $f(x_1, x_2) = x_1 - 1$.

COROLLARY 4.3. *The* LINRF($\mathbb{Z}$) *problem is PTIME-decidable for SLC loops in which the transition polyhedron $\mathcal{Q}$ is guaranteed to be integral. This also applies to any easy class of polyhedra, namely a class where the integer hull is PTIME-computable.*

Proof. Immediate from Lemma 4.1 and the fact that LINRF($\mathbb{Q}$) is PTIME-decidable. □

COROLLARY 4.4. *The* LINRF($\mathbb{Z}$) *problem is PTIME-decidable for SLC loops in which the condition polyhedron $\mathcal{C}$ is guaranteed to be integral, or belongs to an easy class, and the update is linear with integer coefficients.*

Proof. We show that in such case the transition polyhedron $\mathcal{Q}$ is, in fact, integral, and thus Cor. 4.3 applies. Let $\mathcal{C}$ be integral, and the update be $\mathbf{x}' = A'\mathbf{x} + \mathbf{c}'$ where the entries of A' and $\mathbf{c}'$ are integer. Let $\mathbf{x}'' = \binom{\mathbf{x}}{\mathbf{x}'} \in \mathcal{Q}$, that is, $\mathbf{x} \in \mathcal{C}$ and $\mathbf{x}' = A'\mathbf{x} + \mathbf{c}'$. Since $\mathcal{C}$ is integral, $\mathbf{x}$ is a convex combination of some integer points. I.e, $\mathbf{x} = \sum a_i \cdot \mathbf{x}_i$ where $a_i \geq 0$, $\sum a_i = 1$ and $\mathbf{x}_i \in I(\mathcal{C})$. Hence, $\mathbf{x}' = A'(\sum a_i \cdot \mathbf{x}_i) + \mathbf{c}' = \sum a_i \cdot (A'\mathbf{x}_i + \mathbf{c}')$ and

$$\mathbf{x}'' = \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \sum a_i \cdot \mathbf{x}_i \\ \sum a_i \cdot (A'\mathbf{x}_i + \mathbf{c}') \end{pmatrix} = \sum a_i \cdot \begin{pmatrix} \mathbf{x}_i \\ A'\mathbf{x}_i + \mathbf{c}' \end{pmatrix}$$

Now note that $\binom{\mathbf{x}_i}{A'\mathbf{x}_i + \mathbf{c}'}$ are integer points from $I(\mathcal{Q})$, which implies that $\mathbf{x}''$ is a convex combination of integer points in $\mathcal{Q}$. Hence, $\mathcal{Q}$ is integral. □

Corollaries 4.3 and 4.4 suggest looking for classes of SLC loops where we can easily ascertain that $\mathcal{Q}$ is integral, or that its integer hull can be computed in polynomial time. In what follows we address such cases: Sec. 4.1 discusses special cases in which the transition or condition polyhedron is integral by construction; Sec. 4.2 shows that for certain cases of *two-variables per inequality* constraints the integer hull can be computed in a polynomial time; Sec. 4.3 discusses the case of octagonal relations; Sec. 4.4 shows that for some cases LINRF($\mathbb{Z}$) is even strongly polynomial; and Sec. 4.5 extends the results to MLC loops.

4.1 Loops specified by integer polyhedra

There are some well-known examples of polyhedra that are known to be integral due to some structural property. This gives us classes of SLC loops where LINRF($\mathbb{Z}$) is in PTIME. The examples below are all from [41], where the proofs of the lemmas can be found.

LEMMA 4.5. *For any rational matrix B, the* cone $\{\mathbf{x} \mid B\mathbf{x} \leq \mathbf{0}\}$ *is an integer polyhedron.*

COROLLARY 4.6. *The* LINRF($\mathbb{Z}$) *problem is PTIME-decidable for SLC loops of the form*

$$\textit{while } (B\mathbf{x} \leq \mathbf{0}) \textit{ do } \mathbf{x}' = A'\mathbf{x} + \mathbf{c}'$$

where the entries in A' and $\mathbf{c}'$ are integer.

Recall that a matrix A is totally unimodular if each subdeterminant of A is in $\{0, \pm 1\}$. In particular, the entries of such matrix are from $\{0, \pm 1\}$.

LEMMA 4.7. *For any totally unimodular matrix A and integer vector $\mathbf{b}$, the polyhedron $\mathcal{P} = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ is integral.*

For brevity, if a polyhedron $\mathcal{P}$ is specified by $A\mathbf{x} \leq \mathbf{b}$ in which A is a totally unimodular matrix and $\mathbf{b}$ an integer vector, we say that $\mathcal{P}$ is totally unimodular.

COROLLARY 4.8. *The* LINRF($\mathbb{Z}$) *problem is PTIME-decidable for SLC loops in which (1) the transition polyhedron $\mathcal{Q}$ is totally unimodular; or (2) the condition polyhedron $\mathcal{C}$ is totally unimodular and the update is linear with integer coefficients.*

As a notable example, difference bound constraints [7, 10, 11] are defined by totally unimodular matrices. Such constraints have the form $x - y \leq d$ with $d \in \mathbb{Q}$; constraints of the form $\pm x \leq d$ can also be admitted. In the integer case we can always tighten d to $\lfloor d \rfloor$ and thus get an integer polyhedron.

It might be worth mentioning that checking if a matrix is totally unimodular can be done in polynomial time [41, Th. 20.3, p. 290].

4.2 Two-variable per inequality constraints

In this section we consider cases in which the input loop involves *two-variable per inequality* constraints ($TVPI$ for brevity), i.e., inequalities of the form $ax + by \leq d$ with $a, b, d \in \mathbb{Q}$. Clearly, polyhedra defined by such inequalities are not guaranteed to be integral. See, for example, Fig. 1(B).

Harvey [29] showed that for *two-dimensional* polyhedra, which are specified by $TVPI$ constraints by definition, the integer hull can be computed in $O(m \log A_{max})$ where m is the number of inequalities and A_{max} is the magnitude of the largest coefficient.

DEFINITION 4.9. Let T be a set of $TVPI$ constraints. We say that T is a *product of independent two-dimensional $TVPI$ constraints* ($PTVPI$ for brevity), if it can be partitioned into $T_1, \ldots, T_n$ such that (1) each T_i is two-dimensional, i.e., involves at most two variables; and (2) each distinct T_i and T_j do not share variables.

LEMMA 4.10. *The integer hull of $PTVPI$ constraints can be computed in polynomial time.*

Proof. According to [41, Sec. 16.3, p. 231], a polyhedron $\mathcal{P}$ is integral iff each of its *faces* has an integer point. A face of $\mathcal{P}$ is obtained by turning some inequalities to equalities such that the resulting polyhedron in not empty (over the rationals). Clearly, if T_1 and T_2 are two sets of inequalities that do not share variables, and their faces have integer points, then all faces of $T_1 \cup T_2$ have integer points. Thus, $T_1 \cup T_2$ is integral. Partitioning T into independent $T_1, \dots, T_n$ and checking that each is two-dimensional can done in polynomial time. Computing the integer hull of each T_i can be done in polynomial time using Harvey's method. □

The above approach can easily be generalized. Given any polyhedron, we first decompose it into independent sets of inequalities, in polynomial time (these are the connected components of an obvious graph), and then check if each set is covered by one of the special cases for which the integer hull can be efficiently computed.

COROLLARY 4.11. *The* LinRF($\mathbb{Z}$) *problem is PTIME-decidable for SLC loops in which: (1) the transition polyhedron $\mathcal{Q}$ is $PTVPI$; or (2) the condition polyhedron $\mathcal{C}$ is $PTVPI$, and the update is linear with integer coefficients.*

EXAMPLE 4.12. Consider the following SLC loop, as an example for case (1) of Cor. 4.11

$$\begin{array}{l} \textit{while } (4x_1 \geq 1, x_2 \geq 1) \textit{ do} \\ \quad 2x_1 - 5x_1' \leq 3,\ -2x_1 + 5x_1' \leq 1,\ x_2' = x_2 + 1 \end{array} \tag{15}$$

Applying LinRF($\mathbb{Q}$) does not find a LRF since the loop does not terminate when the variables range over $\mathbb{Q}$, e.g., for $x_1 = \frac{1}{4}$ and $x_2 = 1$. The transition polyhedron is not integral, however, it is $PTVPI$ since it can be divided into $T_1 = \{4x_1 \geq 1, 2x_1 - 5x_1' \leq 3, -2x_1 + 5x_1' \leq 1\}$ and $T_2 = \{x_2 \geq 1, x_2' = x_2 + 1\}$. It is easy to check that T_2 is already integral. Computing the integer hull of T_1 adds the inequalities $-x_1 + x_1' \leq -1$ and $\frac{1}{3}x_1 - x_1' \leq \frac{1}{3}$. See Fig. 1(C). Now LinRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2) = x_1 - 1$.

EXAMPLE 4.13. Consider the following loop, as an example for case (2) of Cor. 4.11

$$\begin{array}{l} \textit{while } (-x_1 + x_2 \leq 0, -2x_1 - x_2 \leq -1, x_3 \leq 1) \textit{ do} \\ \quad x_1' = x_1,\ x_2' = x_2 - 2x_1 + x_3,\ x_3' = x_3 \end{array} \tag{16}$$

Applying LinRF($\mathbb{Q}$) does not find a LRF since it does not terminate over $\mathbb{Q}$, e.g., for $x_1 = x_2 = \frac{1}{2}$ and $x_3 = 1$. The condition polyhedron is not integral, but it is $PTVPI$ since the constraints can be divided into $T_1 = \{-x_1 + x_2 \leq 0, -2x_1 - x_2 \leq -1\}$ and $T_2 = \{x_3 \leq 1\}$. It is easily seen that T_2 is already integral; computing the integer hull of T_1 adds $x_1 \geq 1$. See Fig. 1(B). Now LinRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2, x_3) = 2x_1 + x_2 - 1$. Note that the update in this loop involves constraints which are not $TVPI$.

4.3 Octagonal relations

$TVPI$ constraints in which the coefficients are from $\{0, \pm 1\}$ have received considerable attention in the area of program analysis. Such constraints are called *octagonal relations* [37]. A particular interest was in developing efficient algorithms for checking satisfiability of such relations, as well as inferring all implied *octagonal* inequalities, for variables ranging either over $\mathbb{Q}$ or over $\mathbb{Z}$.

Over $\mathbb{Q}$, this is done by computing the *transitive closure* of the relation, which basically adds inequalities that result from the addition of two existing inequalities, and possibly scaling to obtain coefficients of ± 1. E.g., starting from the set of inequalities $\{-x_1 + x_2 \leq 0, -x_1 - x_2 \leq -1\}$, we add $-2x_1 \leq -1$, or, after scaling, $-x_1 \leq -\frac{1}{2}$. Over $\mathbb{Z}$, this is done by computing the *tight closure*, which in addition to transitivity, is closed also under tightening. This operation replaces $ax + by \leq d$ by $ax + by \leq \lfloor d \rfloor$. For example, tightening $-x_1 \leq -\frac{1}{2}$ yields $-x_1 \leq -1$. The *tight closure* can be computed in polynomial time [4, 30, 39]. Since the tightening eliminates some non-integer points, it is tempting to expect that it actually computes the integer hull. It is easy to show that this is true for two-dimensional relations, but it is false already in three dimensions, as we show in the following example.

EXAMPLE 4.14. Consider the following loop

$$\begin{array}{l} \textit{while } (x_1 + x_2 \leq 2, x_1 + x_3 \leq 3, x_2 + x_3 \leq 4) \textit{ do} \\ \quad x_1' = 1 - x_1, x_2' = 1 + x_1, x_3' = 1 + x_2 \end{array} \tag{17}$$

Note that the transition polyhedron is octagonal, but not integral. Applying LinRF($\mathbb{Q}$) does not find a LRF, since the loop does not terminate over $\mathbb{Q}$, e.g., for $x_1 = \frac{1}{2}$, $x_2 = \frac{3}{2}$, and $x_3 = \frac{5}{2}$. Computing the tight closure does not change the transition (or condition) polyhedron, and thus, it is of no help in finding the LRF. In order to obtain the integer hull of the transition (or condition) polyhedron we should add $x_1 + x_2 + x_3 \leq 4$, which is not an octagonal inequality. Having done so, LinRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2, x_3) = -3x_1 - 4x_2 - 2x_3 + 12$.

Although it is not guaranteed that the tight closure of an octagonal relation corresponds to its integer hull, in practice, it does in many cases. Thus, since it can be computed in polynomial time, we suggest computing it before applying LinRF($\mathbb{Q}$) on loops that involve such relations. The above example shows that this does not give us a complete polynomial-time algorithm for LinRF($\mathbb{Z}$) over octagonal relations.

EXAMPLE 4.15. Consider the following SLC loop

$$\textit{while } \left(\begin{array}{ll} -x_1 + x_2 \leq 0, & -x_1 - x_2 \leq -1, \\ x_2 - x_3 \leq 0, & -x_2 - x_3 \leq -1 \end{array} \right) \textit{ do}$$
$$x_1' = x_1,\ x_2' = x_2 - x_1 - x_3 + 1,\ x_3' = x_3$$

The condition polyhedron is octagonal, but not integral; moreover, it is not $PTVPI$. LinRF($\mathbb{Q}$) does not find a LRF (indeed the loop fails to terminate for $x_1 = x_2 = x_3 = \frac{1}{2}$). Computing the tight closure of the condition adds $-x_1 \leq -1$ and $-x_3 \leq 0$, which results in the integer hull. Now LinRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2, x_3) = x_1 + x_2 - 1$.

A polynomial-time algorithm for computing the integer hull of octagonal relations is, unfortunately, ruled out by examples of such relations whose integer hulls have exponentially many facets.

THEOREM 4.16. *There is no polynomial-time algorithm for computing the integer hull of general octagonal relations.*

Proof. We build an octagonal relation $\mathcal{O}$, such that the minimum number of inequalities required to describe its integer hull $\mathcal{O}_I$ is not polynomial in the number of inequalities in $\mathcal{O}$.

For a complete graph $K_n = \langle V, E \rangle$, we let $\mathcal{P}$ be defined by the set of inequalities $\{x_e \geq 0 \mid e \in E\} \cup \{\sum_{v \in e} x_e \leq 1 \mid v \in V\}$. Here every edge $e \in E$ has a corresponding variable x_e, and the notation $v \in e$ means that v is a vertex of edge e. Note that $\mathcal{P}$ is not octagonal. It is well-known that $\mathcal{P}_I$, the matching polytope of K_n, has at least $\binom{n}{2} + 2^{n-1}$ facets [41, Sec. 18.2, p. 251], and thus any set of inequalities that defines $\mathcal{P}_I$ must have at least the same number of inequalities. Now let $\mathcal{O}$ be defined by $\{x_e \geq 0 \mid e \in E\} \cup \{x_{e_1} + x_{e_2} \leq 1 \mid v \in e_1, v \in e_2\}$, which includes $n + n \cdot \binom{n-1}{2}$ *octagonal inequalities*. It is easy to see that the integer solutions of $\mathcal{P}$ and $\mathcal{O}$ are the same, and thus $\mathcal{P}_I = \mathcal{O}_I$. This means that any set of inequalities that define $\mathcal{O}_I$ must have at least $\binom{n}{2} + 2^{n-1}$ inequalities. Therefore, any algorithm that computes such a representation must add at least $\binom{n}{2} + 2^{n-1} - n - n \cdot \binom{n-1}{2}$ inequalities to $\mathcal{O}$, which is super-polynomial in the size of $\mathcal{O}$. Unsurprisingly, the tight closure of $\mathcal{O}$ does not yield its integer hull (it only adds $x_e \leq 1$ for each x_e). □

Note that the above theorem does not rule out a polynomial-time algorithm for LINRF($\mathbb{Z}$), for SLC loops in which the transition polyhedron $\mathcal{Q}$ is octagonal, or where the condition polyhedron is octagonal and the update is linear with integer coefficients. It just rules out an algorithm that is based on computing the integer hull of the polyhedra. However, the coNP-hardness proof of Sec. 3.1 could be also carried out by a reduction from 3SAT that produces an SLC loop where the condition is octagonal and the update is linear with integer coefficients—so at least for this class there is, presumably, no polynomial solution. This reduction can be found in the technical report [8].

4.4 Strongly polynomial cases

Polynomial-time algorithms for LINRF($\mathbb{Q}$) [3, 36, 38] inherit their complexity from that of LP. While it is known that LP can be solved by a polynomial-time algorithm, it is an open problem whether it has a *strongly polynomial* algorithm. Such an algorithm should perform a number of elementary arithmetic operations polynomial in the *dimensions* of the input matrix instead of its bit-size (which accounts for the size of the matrix entries), and such operations should be performed on numbers of size which is polynomial to the input bit-size. However, there are some cases for which LP is known to have a strongly polynomial algorithm. We first use these cases to define classes of SLC loops for which LINRF($\mathbb{Q}$) has a strongly polynomial algorithm, which we then use to show that LINRF($\mathbb{Z}$) has a strongly polynomial algorithm for some corresponding classes of SLC loops. Our results are based on the following result by Tardos [44] (quoted from [41, p. 196]).

THEOREM 4.17 (Tardos)**.** *There exists an algorithm which solves a given rational LP problem $\max\{\mathbf{c}\cdot\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$ with at most $P(size(A))$ elementary arithmetic operations on numbers of size polynomially bounded by $size(A, \mathbf{b}, \mathbf{c})$, for some polynomial P.*

Note that the number of arithmetic operations required by the LP algorithm only depends on the bit-size of A. Clearly, if we restrict the LP problem to cases in which the bit-size of the entries of A is bounded by a constant, then $size(A)$ depends only on its dimensions, and we get a strongly polynomial time algorithm. In particular we can state the following.

COROLLARY 4.18. *There exists a strongly polynomial algorithm to solve an LP problem $\max\{\mathbf{c}\cdot\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$ where the entries of A are $\{0, \pm1, \pm2\}$.*

We can use this to show that LINRF($\mathbb{Q}$) can sometimes be implemented with strongly polynomial complexity. To do this, we use the Podelski-Rybalchenko formulation of the procedure [38], slightly modified to require that the LRF decreases at least by 1 instead of by some $\delta > 0$.

THEOREM 4.19 (Podelski-Rybalchenko)**.** *Given an SLC loop with a transition polyhedron $\mathcal{Q} \subseteq \mathbb{Q}^{2n}$, specified by $A''\mathbf{x}'' \le \mathbf{c}''$, let $A'' = (A\ A')$ where each A and A' has n columns and m rows, and let $\vec{\mu}, \vec{\eta}$ be row vectors of different m rational variables each. A LRF for $\mathcal{Q}$ exists iff there is a (rational) solution to the following set of constraints*

$$\vec{\mu}, \vec{\eta} \ge \mathbf{0}^T, \quad (18a) \qquad \vec{\eta}\cdot(A + A') = \mathbf{0}^T, \quad (18d)$$

$$\vec{\mu}\cdot A' = \mathbf{0}^T, \quad (18b) \qquad \vec{\eta}\cdot\mathbf{c}'' \le -1\,. \quad (18e)$$

$$(\vec{\mu} - \vec{\eta})\cdot A = \mathbf{0}^T, \quad (18c)$$

THEOREM 4.20. *The* LINRF($\mathbb{Q}$) *problem is decidable in strongly polynomial time for SLC loops specified by $A''\mathbf{x}'' \le \mathbf{c}''$ where the coefficients of A'' are from $\{0, \pm1\}$.*

Proof. First observe that, in Th. 4.19, when the matrix A'' has only entries from $\{0, \pm1\}$, then all coefficients in the constraints (18a–18d) are from $\{0, \pm1, \pm2\}$. Moreover, the number of inequalities and variables in (18a–18d) is polynomial in the dimensions of A''. Now let us modify the Podelski-Rybalchenko procedure such that instead of testing for feasibility of the constraints (18a–18e), we consider the minimization of $\vec{\eta}\cdot\mathbf{c}''$ under the other constraints (18a–18d). Clearly, this answers the same question since: (18a–18e) is feasible, iff the minimization problem is unbounded, or the minimum is negative. This brings the problem to the form required by Cor. 4.18 and yields our result. ☐

COROLLARY 4.21. *The* LINRF($\mathbb{Z}$) *is decidable in strongly polynomial time for SLC loops, specified by $A''\mathbf{x}'' \le \mathbf{c}''$, that are covered by any of the special cases of secs. 4.1 and 4.2 and the entries of A'' are from $\{0, \pm1\}$.*

Proof. In the cases of Sec. 4.1, the transition polyhedron is guaranteed to be integral. In the $PTVPI$ case of Sec. 4.2: (1) the integer hull can be computed using Harvey's procedure, which is strongly polynomial in this case since the entries of A are from $\{0, \pm1\}$. This can be done also using the tight closure of 2-dimensional octagons; and (2) the $TVPI$ constraints that we add when computing the integer hull have coefficients from $\{0, \pm1\}$, and the number of such constraints is polynomially bounded by the number of the original inequalities. Thus, by Th. 4.20, we can apply a strongly polynomial-time algorithm for LINRF($\mathbb{Q}$). ☐

4.5 Multipath loops

Recall that a linear function ρ is a LRF for an MLC loop with transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, iff it is a LRF for each $\mathcal{Q}_i$. Thus, if we have the set of $LRFs$ for each $\mathcal{Q}_i$, we can simply take the intersection and obtain the set of $LRFs$ for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$. In the Podelski-Rybalchenko procedure, the set of solutions for the inequalities (18a–18e) defines the set of $LRFs$ for the corresponding SLC loop as follows.

LEMMA 4.22. *Given an SLC loop with a transition polyhedron $\mathcal{Q}$, specified by $A''\mathbf{x}'' \le \mathbf{c}''$, let $\Delta(\vec{\mu}, \vec{\eta}, A'', c'')$ be the conjunction of (18a–18e). Then, $\rho(\mathbf{x}) = \vec{\lambda}\cdot\mathbf{x} + \lambda_0$ is a LRF for $\mathcal{Q}$ iff $\Delta(\vec{\mu}, \vec{\eta}, A'', c'')$ has a solution such that $\vec{\lambda} = \vec{\eta}\cdot A'$ and $\lambda_0 \ge \vec{\mu}\cdot\mathbf{c}''$.*

The following lemma shows how to compute, using the above lemma, the intersection of sets of $LRFs$ for several transition polyhedra, and thus obtain the set of $LRFs$ for a given MLC loop. (a very similar statement appears in [22, Lemma 3]).

LEMMA 4.23. *Given an MLC loop with transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, each specified by $A''_i\mathbf{x}'' \le \mathbf{c}''_i$, let $\Delta(\vec{\mu}_i, \vec{\eta}_i, A''_i, c''_i)$ be the constraints (18a–18e) for the i-th path, and $(\lambda_0, \vec{\lambda})$ be $n+1$ rational variables. Then, there is a LRF for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, iff the following is feasible (over the rationals)*

$$\bigwedge_{i=1}^{k} \Delta(\vec{\mu}_i, \vec{\eta}_i, A''_i, c''_i) \wedge \vec{\lambda} = \vec{\eta}_i\cdot A'_i \wedge \lambda_0 \ge \vec{\mu}_i\cdot\mathbf{c}''_i \qquad (19)$$

Moreover, the values of $(\lambda_0, \vec{\lambda})$ in the solutions of (19) *define the set of all $LRFs$ for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$.*

Proof. Immediate by Lemma 4.22, noting that for each i the constraints $\Delta(\vec{\mu}_i, \vec{\eta}_i, A''_i, c''_i)$ use different $\vec{\mu}_i$ and $\vec{\eta}_i$, while $(\lambda_0, \vec{\lambda})$ are the same for all i. ☐

COROLLARY 4.24. *The* LINRF($\mathbb{Q}$) *problem for MLC loops is PTIME-decidable.*

Proof. The size of the set of inequalities (19) is polynomial in the size of the input MLC loop, and checking if it has a rational solution can be done in polynomial time. ☐

COROLLARY 4.25. *The* LINRF($\mathbb{Z}$) *problem for MLC loops is PTIME-decidable when each path corresponds to one of the special cases, for SLC loops, discussed in secs. 4.1 and 4.2.*

Proof. Immediate, since if the transition polyhedra are integral, LINRF($\mathbb{Z}$) and LINRF($\mathbb{Q}$) are equivalent. □

EXAMPLE 4.26. Consider an MLC loop with the following two paths: Loop (1) of Sec. 1; and the loop of Ex. 4.12. Applying LINRF($\mathbb{Q}$) (as in Lemma 4.23) does not find a LRF since both paths do not terminate when the variables range over $\mathbb{Q}$. If we first compute the integer hull of both paths, LINRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2) = 3x_1 + x_2 - 2$. Note that the integer hull of the first path is computable in polynomial-time since the condition is $PTVPI$ and the update is linear with integer coefficients. That of the second path is also computable in polynomial-time as in Ex. 4.12.

5. Prototype Implementation

The different algorithms presented in this paper, both for the general case and the special PTIME cases, have been implemented. The tool receives as input an MLC loop in constraints representation, and allows applying LINRF($\mathbb{Z}$) or LINRF($\mathbb{Q}$). It can be tried out via `http://www.loopkiller.com/irankfinder`, where all examples of this paper are also available. The implementation includes the algorithms of Lemmas 3.19 and 4.23. By default it uses the second one since the first one requires generating the generator representation of the transition polyhedron, which is exponential.

Computing the integer hull of a polyhedron, in the case of LINRF($\mathbb{Z}$), is done by first decomposing its set of inequalities into independent sets, and then computing the integer hull of each set separately. Each set of inequalities is first matched against one of the PTIME cases of secs. 4.1 and 4.2. If this matching fails, it computes the integer hull using Hartmann's algorithm [28] as explained by Charles et al. [18]. Note that this algorithm supports only bounded polyhedra, the integer hull of an unbounded polyhedron is computed by considering a corresponding bounded one [41, Th. 16.1, p. 231]. In addition, for octagonal relations, it gives the possibility of computing the tight closure instead of the integer hull. As we have seen in Sec. 4.3, when this option is used, completeness of LINRF($\mathbb{Z}$) is not guaranteed. The Parma Polyhedra Library [5] is used for converting between generator and constraints representations, solving (mixed) LP problems, etc.

6. Related work

There are several works [3, 20, 36, 38, 42] that directly address the LINRF($\mathbb{Q}$) problem for SLC or MLC loops. In all these works, the underlying techniques allow synthesizing $LRFs$ and not only deciding if one exists. The common observation to all these works is that synthesising $LRFs$ can be done by inferring the implied inequalities of a given polyhedron (the transition polyhedron of the loop), in particular inequalities like conditions (4) and (5) of Def. 2.3 that define a LRF. Regarding completeness, all these methods are complete for LINRF($\mathbb{Q}$) but not for LINRF($\mathbb{Z}$). They can also be used to approximate LINRF($\mathbb{Z}$) by relaxing the loop such that its variables range over $\mathbb{Q}$ instead of $\mathbb{Z}$, thus sacrificing completeness. All these methods have a corresponding PTIME algorithm. Exceptions in this line of research are the work of Bradley et al. [13] and Cook et al. [22] that directly address the LINRF($\mathbb{Z}$) problem for MLC loops.

Sohn and Van Gelder [42] considered MLC loops with variables ranging over $\mathbb{N}$. These are abstractions of loops from logic programs. The loops were, in fact, relaxed from $\mathbb{N}$ to $\mathbb{Q}_+$ before seeking a LRF, however, this is not explicitly mentioned. The main observation is that the duality theorem of LP [41, p. 92] can be used to infer inequalities that are implied by the transition polyhedron. The authors also mention that this was observed before by Lassez [33] in the context of solving CLP($\mathbb{R}$) queries. Completeness was not addressed in this work, and the PTIME complexity was mentioned but not formally addressed. Later, Mesnard and Serebrenik [36] formally proved that the techniques of Sohn and Van Gelder [42] provide a complete PTIME method for LINRF($\mathbb{Q}$), also for the case of MLC loops. They pointed out the incompleteness for LINRF($\mathbb{Z}$).

Probably the most popular work on the synthesis of $LRFs$ is the one of Podelski and Rybalchenko [38]. They also observed the need for deriving inequalities implied by the transition polyhedron, but instead of using the duality theorem of LP they used the affine form of Farkas' lemma [41, p. 93]. Completeness was claimed, and the statement did not make it clear that the method is complete for LINRF($\mathbb{Q}$) but not for LINRF($\mathbb{Z}$). This was clarified, however, in the PhD thesis of Rybalchenko [40].

Bagnara et al. [6] proved that the methods of Sohn and Van Gelder [36] and Podelski and Rybalchenko [38] are actually equivalent, i.e., they compute the same set of $LRFs$. They also showed that the method of Podelski and Rybalchenko can, potentially, be more efficient since it requires solving rational constraints systems with fewer variables and constraints.

The earliest appearances of a solution based on Farkas' Lemma, that we know of, are by Colón and Sipma [20], in the context of termination analysis, and by Feautrier [25], in the context of automatic parallelization of computations. Colón and Sipma did not claim that the problem can be solved in polynomial time, and indeed their implementation seems to have exponential complexity since they use generators and polars, despite the similarity of the underlying theory to that of [38]. Completeness was claimed, however it was not explicitly mentioned that the variables range over $\mathbb{Q}$ and not $\mathbb{Z}$.

Feautrier [25] described scheduling of computations that can be described by recursive equations. An abstraction to a form similar to an MLC loop allowed him to compute a so-called *schedule*, which is essentially a ranking function, but used backwards, since the computations at the bottom of the recursion tree are to be completed first.

Cook et al. [22] observed that the Farkas-lemma based solution is complete for LINRF($\mathbb{Z}$) when the input MLC loop is specified by integer polyhedra. They also mention that any polyhedron can be converted to an integer one, and that this might increase its size exponentially. Unlike our work, they do not address PTIME cases or the complexity of LINRF($\mathbb{Z}$). In fact, the main issue in that work is the synthesis of ranking functions for bit-victors relations.

Bradley et al. [13] directly addressed the LINRF($\mathbb{Z}$) problem for MLC loops, and stated that the methods of Colón and Sipma [20] and Podelski and Rybalchenko [38] are not complete for LINRF($\mathbb{Z}$). Their technique is based on the observation that if there is a LRF, then there exists one in which each coefficient λ_i has a value in the interval $[-1, 1]$, and moreover with denominators that are power of 2. Using this observation, they recursively search for the coefficients starting from a region defined by a hyper-rectangle in which each λ_i is in the interval $[-1, 1]$. Given a hyper-rectangle, the algorithm first checks if one of its corners defines a LRF, in which case it stops. Otherwise, the region is either pruned (if it can be verified that it contains no solution), or divided into smaller regions for recursive search. Testing if a region should be pruned is done by checking the satisfiability of a possibly exponential (in the number of variables) number of Presburger formulas. The algorithm will find a LRF if exists, but it might not terminate if no LRF exists. To make it practical, it is parametrized by the search depth, thus sacrificing completeness. It is interesting to note that the search-depth parameter in their algorithm actually bounds the bit-size of the LRF coefficients. Our Cor. 3.21 shows that it is possible to deterministically bound this depth, that turning their algorithm into a complete one, though still exponential.

Codish et al. [19] studied the synthesis of $LRFs$ for SLC loops with *size-change* constraints (i.e., of the form $x_i \geq x'_j + c$ where $c \in \{0, 1\}$), and *monotonicity* constraints (i.e., of the form $X \geq Y + c$, where X and Y are variables or primed variables, and $c \in \{0, 1\}$). In both cases the variables ranged over $\mathbb{N}$. For size-change constraints, they proved that the loop terminates iff a LRF exists, moreover, such function has the form $\sum \lambda_i \cdot x_i$ with $\lambda_i \in \{0, 1\}$. For the case of monotonicity constraints, they proved that the loop terminates iff a LRF exists for the *balanced* version of the loop, and has the form $\sum \lambda_i \cdot x_i$ with $\lambda_i \in \{0, \pm 1\}$. Intuitively, a balanced loop includes $x'_i \geq x'_j + c$ iff it includes $x_i \geq x_j + c$. They showed how to balance the loop while preserving its termination behavior. Recently, Bozga et al. [10] presented similar results for SLC loops defined by octagonal relations.

Bradley et al. [12] extended the work of Colón and Sipma [20] and searched *lexicographic* $LRFs$. In [14, 15] they considered multipath loops with polynomial transitions and the synthesis of lexicographic polynomial ranking functions, where the notion of ranking functions was also relaxed to functions that *eventually* decrease. Cousot [23] used Lagrangian relaxation for inferring possibly non-linear ranking functions. In the linear case, Lagrangian relaxation is similar to the affine form of Farkas' lemma.

Alias et al. [3] again rediscovered the Farkas-lemma based solution for LINRF($\mathbb{Q}$), or rather "ported" it from the former work by Feautrier, but this time for termination and cost analysis. Like [12], they construct lexicographic $LRFs$, however, they do it for programs with an arbitrary control-flow graph, and they prove completeness of their procedure. Their goal was to use these functions to derive cost bounds (like a bound on the worst-case number of transitions in terms of the initial state); this bound is (when it can be found) a polynomial, whose degree is at most the dimension of the (co-domain of the) lexicographic ranking function. Alias et al.'s construction produces a function of minimum dimension. They, too, have relaxed the problem from integers to rationals and failed to state that their completeness results depend on this relaxation.

Decidability and complexity of termination (in general, not necessarily with $LRFs$) of SLC and MLC loops has been intensively studied for different classes of constraints. For SLC loops, Tiwari [45] proved that the problem is decidable when the update is linear and the variables range over $\mathbb{R}$. Braverman [16] proved that this holds also for $\mathbb{Q}$, and for the homogeneous case it holds for $\mathbb{Z}$. Both considered universal termination. Also, in both cases they allow the use of strict inequalities in the condition. Ben-Amram et al. [9] showed that the termination of SLC loops is at least EXPSPACE-hard, and that the problem is undecidable for some extensions that introduce a simple form of non-linearity; and also for SLC loops in which the use of a single irrational coefficient is allowed. See these works for references to additional results on the decidability of termination in other types of loops.

7. Concluding remarks

We have studied the Linear Ranking problem for single-path and multipath linear constraint loops and observed the difference between the LINRF($\mathbb{Q}$) problem, where variables range over the rationals, and the LINRF($\mathbb{Z}$) problem, where variables only take integer values. In practice, the latter is more common, but the complexity of the problem has not been studied before; the common approach has been to relax the problem to the rationals, where complete, polynomial-time decision procedures have been known.

We have confirmed that LINRF($\mathbb{Z}$) is a harder problem, proving it to be coNP-complete. On a positive note, this shows that there *is* a complete solution, even if exponential-time. We further showed that some special cases of importance do have a PTIME solution.

The latter results arise from a proof that for integer polyhedra, LINRF($\mathbb{Z}$) and LINRF($\mathbb{Q}$) are equivalent. Interestingly, this is not the case for termination in general. For example, the transition polyhedron of the loop "$while\ x \geq 0\ do\ x' = 10 - 2x$" is integral; the loop terminates when the variables range over $\mathbb{Z}$ but does not terminate when they range over $\mathbb{Q}$, specifically for $x = 3\frac{1}{3}$. Note that this loop does not have a LRF over the integers.

A more general notion of ranking function applies to an arbitrary control-flow graph with transitions specified by source and target nodes as well as linear constraints on the values of variables. In this setting, one seeks to associate a (possibly different) affine function ρ_ν with each node ν, so that on a transition from ν to ν' we have $\rho_\nu(\mathbf{x}) > \rho_{\nu'}(\mathbf{x}')$. Such functions can be found by LP, a procedure complete over the rationals, using a simple extension of the solution for the loops we have discussed [3, 36]. The considerations regarding the complexity of the corresponding problem over integers are essentially the same as those we have presented, and we preferred to use the simpler model for clearer presentation.

In all examples that we have discussed in this paper, when a loop has a LRF over $\mathbb{Z}$ but not over $\mathbb{Q}$, then the loop did not terminate over $\mathbb{Q}$. This is, however, not the case in general. A counter-example can be constructed by combining (i.e., executing simultaneously) the loop of Ex. 3.6 and Loop (1) of Sec. 1.

In the context of complexity (cost) analysis, there is a special interest in $LRFs$ that decrease at least by 1 in each iteration, since they bound the number of iterations of a given loop. In order to get tight bounds, even if $\mathcal{Q}$ has a LRF it might be worthwhile to compute one for $I(\mathcal{Q})$. To see this, let us add $4x_1 \geq 3$ to the condition of Loop (1) in Sec. 1. Then, both $\mathcal{Q}$ and $I(\mathcal{Q})$ have $LRFs$. For $I(\mathcal{Q})$ the most tight one (under the requirement to decrease by at least 1) is $f_1(x_1, x_2) = x_1 + x_2 - 1$, while for $\mathcal{Q}$ it is $f_2(x_1, x_2) = 2x_1 + 2x_2 - 2$. Hence, a better bound is obtained using $I(\mathcal{Q})$. The same observation applies to loop parallelization: the functions' value gives the schedule's *latency* (depth of the computation tree) and a lower value is preferable.

Acknowledgments

We thank Alain Darte for a helpful discussion. The work of Samir Genaim was funded in part by the EU project ICT-231620 *HATS*, and the Spanish projects TIN-2008-05624 *DOVES* and TIN-2012-38137 *VIVAC*.

References

[1] E. Albert, P. Arenas, S. Genaim, G. Puebla, and D. Zanardini. Costa: Design and implementation of a cost and termination analyzer for java bytecode. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W. P. de Roever, editors, *Formal Methods for Components and Objects, FMCO'07*, volume 5382 of *LNCS*, pages 113–132. Springer, 2007.

[2] E. Albert, P. Arenas, S. Genaim, and G. Puebla. Closed-form upper bounds in static cost analysis. *J. Autom. Reasoning*, 46(2):161–203, 2011.

[3] C. Alias, A. Darte, P. Feautrier, and L. Gonnord. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In R. Cousot and M. Martel, editors, *Static Analysis Symposium, SAS'10*, volume 6337 of *LNCS*, pages 117–133. Springer, 2010.

[4] R. Bagnara, P. M. Hill, and E. Zaffanella. An improved tight closure algorithm for integer octagonal constraints. In F. Logozzo, D. Peled, and L. D. Zuck, editors, *Verification, Model Checking, and Abstract Interpretation, VMCAI'08*, volume 4905 of *LNCS*, pages 8–21. Springer, 2008.

[5] 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. *Sci. Comput. Program.*, 72(1-2):3–21, 2008.

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

[7] A. M. Ben-Amram. Size-change termination with difference constraints. *ACM Trans. Program. Lang. Syst.*, 30(3), 2008.

[8] A. M. Ben-Amram and S. Genaim. On the linear ranking problem for integer linear-constraint loops. *CoRR*, abs/1208.4041, 2012.

[9] A. M. Ben-Amram, S. Genaim, and A. N. Masud. On the termination of integer loops. In V. Kuncak and A. Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation, VMCAI'12*, volume 7148 of *LNCS*, pages 72–87. Springer, 2012.

[10] 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, TACAS'12*, volume 7214 of *LNCS*, pages 252–266. Springer, 2012.

[11] L. Bozzelli and S. Pinchinat. Verification of gap-order constraint abstractions of counter systems. In V. Kuncak and A. Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation, VMCAI'12*, volume 7148 of *LNCS*, pages 88–103. Springer, 2012.

[12] A. R. Bradley, Z. Manna, and H. B. Sipma. Linear ranking with reachability. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, CAV'05*, volume 3576 of *LNCS*, pages 491–504. Springer, 2005.

[13] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination analysis of integer linear loops. In M. Abadi and L. de Alfaro, editors, *Concurrency Theory, CONCUR 2005*, volume 3653 of *LNCS*, pages 488–502. Springer, 2005.

[14] 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, *International Colloquium on Automata, Languages and Programming, ICALP'05*, volume 3580 of *LNCS*, pages 1349–1361. Springer, 2005.

[15] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination of polynomial programs. In R. Cousot, editor, *Verification, Model Checking, and Abstract Interpretation, VMCAI'05*, volume 3385 of *LNCS*, pages 113–129. Springer, 2005.

[16] M. Braverman. Termination of integer linear programs. In T. Ball and R. B. Jones, editors, *Computer Aided Verification, CAV'06*, volume 4144 of *LNCS*, pages 372–385. Springer, 2006.

[17] M. Bruynooghe, M. Codish, J. P. Gallagher, S. Genaim, and W. Vanhoof. Termination analysis of logic programs through combination of type-based norms. *ACM Trans. Program. Lang. Syst.*, 29(2), 2007.

[18] P. J. Charles, J. M. Howe, and A. King. Integer polyhedra for program analysis. In A. V. Goldberg and Y. Zhou, editors, *Algorithmic Aspects in Information and Management, AAIM'09*, volume 5564 of *LNCS*, pages 85–99. Springer, 2009.

[19] M. Codish, V. Lagoon, and P. J. Stuckey. Testing for termination with monotonicity constraints. In M. Gabbrielli and G. Gupta, editors, *International Conference on Logic Programming, ICLP'05*, volume 3668 of *LNCS*, pages 326–340. Springer, 2005.

[20] M. Colón and H. Sipma. Synthesis of linear ranking functions. In T. Margaria and W. Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'01*, volume 2031 of *LNCS*, pages 67–81. Springer, 2001.

[21] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In M. I. Schwartzbach and T. Ball, editors, *Programming Language Design and Implementation, PLDI'06*, pages 415–426. ACM, 2006.

[22] B. Cook, D. Kroening, P. Rümmer, and C. M. Wintersteiger. Ranking function synthesis for bit-vector relations. In J. Esparza and R. Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS'10*, volume 6015 of *LNCS*, pages 236–250. Springer, 2010.

[23] P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In R. Cousot, editor, *Verification, Model Checking, and Abstract Interpretation, VMCAI'05*, volume 3385 of *LNCS*, pages 1–24, 2005.

[24] A. Darte. Understanding loops: The influence of the decomposition of Karp, Miller, and Winograd. In *Formal Methods and Models for Codesign, MEMOCODE'10*, pages 139–148. IEEE Computer Society, 2010.

[25] P. Feautrier. Some efficient solutions to the affine scheduling problem. I. one-dimensional time. *International Journal of Parallel Programming*, 21(5):313–347, 1992.

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W.H. Freeman and Co., New York, 1979.

[27] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with aprove. In V. van Oostrom, editor, *Rewriting Techniques and Applications, RTA'04*, volume 3091 of *LNCS*, pages 210–220. Springer, 2004.

[28] M. E. Hartmann. *Cutting Planes and the Complexity of the Integer Hull*. PhD thesis, School of Operations Research and Industrial Engineering, Cornell University, 1988.

[29] W. Harvey. Computing two-dimensional integer hulls. *SIAM J. Comput.*, 28(6):2285–2299, 1999.

[30] W. Harvey and P. J. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Australasian Computer Science Conference, ACSC'97*, pages 102–111, 1997.

[31] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[32] R. M. Karp and C. H. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *Symp. on Foundations of Computer Science, FOCS'80*, pages 1–9. IEEE Computer Society, 1980.

[33] J.-L. Lassez. Querying constraints. In *Symposium on Principles of Database Systems*, pages 288–298. ACM Press, 1990.

[34] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In C. Hankin and D. Schmidt, editors, *Symposium on Principles of Programming Languages, POPL'01*, pages 81–92, 2001.

[35] N. Lindenstrauss and Y. Sagiv. Automatic termination analysis of Prolog programs. In L. Naish, editor, *International Conference on Logic Programming, ICLP'97*, pages 64–77. MIT Press, 1997.

[36] F. Mesnard and A. Serebrenik. Recurrence with affine level mappings is p-time decidable for clp(r). *TPLP*, 8(1):111–119, 2008.

[37] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, Mar. 2006.

[38] 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, VMCAI'04*, volume 2937 of *LNCS*, pages 239–251. Springer, 2004.

[39] P. Z. Revesz. Tightened transitive closure of integer addition constraints. In V. Bulitko and J. C. Beck, editors, *Symposium on Abstraction, Reformulation, and Approximation, SARA'09*, 2009.

[40] A. Rybalchenko. *Temporal Verification with Transition Invariants*. PhD thesis, Universität des Saarlandes, 2004.

[41] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1986.

[42] K. Sohn and A. V. Gelder. Termination detection in logic programs using argument sizes. In D. J. Rosenkrantz, editor, *Symposium on Principles of Database Systems*, pages 216–226. ACM Press, 1991.

[43] F. Spoto, F. Mesnard, and É. Payet. A termination analyzer for java bytecode based on path-length. *ACM Trans. Program. Lang. Syst.*, 32 (3), 2010.

[44] Éva. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34:250–256, 1986.

[45] A. Tiwari. Termination of linear programs. In R. Alur and D. Peled, editors, *Computer Aided Verification, CAV'04*, volume 3114 of *LNCS*, pages 387–390. Springer, 2004.