Ranking Functions for Linear-Constraint Loops

Amir M. Ben-Amram, The Academic College of Tel-Aviv Yaffo
Samir Genaim, Complutense University of Madrid

In this paper we study the complexity of the problems: given a loop, described by linear constraints over a finite set of variables, is there a linear or lexicographical-linear ranking function for this loop? While existence of such functions implies termination, these problems are not equivalent to termination. When the variables range over the rationals (or reals), it is known that both problems are PTIME decidable. However, when they range over the integers, whether for single-path or multipath loops, the complexity has not yet been determined. We show that both problems are coNP-complete. However, we point out some special cases of importance of PTIME complexity. We also present complete algorithms for synthesizing linear and lexicographical-linear ranking functions, both for the general case and the special PTIME cases. Moreover, in the rational setting, our algorithm for synthesizing lexicographical-linear ranking functions extends existing ones, because our definition for such functions is more general, yet it has polynomial-time complexity.

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.

Additional Key Words and Phrases: Integer loops, Termination, Linear constraints.

ACM Reference Format:
ACM V, N, Article A (January YYYY), 54 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Termination analysis has received considerable attention and nowadays several powerful tools for the automatic termination analysis of different programming languages and computational models exist [Giesl et al. 2004; Cook et al. 2006; Albert et al. 2007; Spoto et al. 2010; Kroening et al. 2010; Harris et al. 2011]. 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:

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

where primed variables represent the values at the completion of an iteration. When the variables are modified in the loop body so that they are not affine linear functions of the old ones, the effect is sometimes captured (or approximated) using *linear*

Work of Samir Genaim was funded partially by the EU project FP7-ICT-610582 ENVISAGE (http://www.envisage-project.eu) and by the Spanish projects TIN2008-05624 and TIN2012-38137.

DOI:http://dx.doi.org/10.1145/0000000.0000000

constraints. For example, the C loop "`while (4*x1>=x2 && x2>=1) x1=(2*x1+1)/5;`", which involves integer division, can be represented by linear constraints as follows (since `2*x1+1` is always positive)

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

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. [Lindenstrauss and Sagiv 1997; Lee et al. 2001; Bruynooghe et al. 2007; Spoto et al. 2010; Magill et al. 2010]. For a precise definition of the loop representations we consider, see Section 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 $\rho(x_1, \dots, x_n) = a_1x_1 + \cdots + a_nx_n + a_0$, with the rationals as a co-domain, such that

(i) $\rho(\bar{x}) \geq 0$ for any valuation $\bar{x}$ that satisfies the loop guard; and
(ii) $\rho(\bar{x}) - \rho(\bar{x}') \geq 1$ for any transition (single execution of the loop body) that starts in $\bar{x}$ and leads to $\bar{x}'$.

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. $a \succeq b$ if and only if $a \geq b + 1$ (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 [Feautrier 1992a; Sohn and Gelder 1991; Colón and Sipma 2001; Podelski and Rybalchenko 2004a; Mesnard and Serebrenik 2008; Alias et al. 2010]. 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, 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 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 Section 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 synthesis algorithm is based on first computing the integer hull of the transition polyhedron defined by the loop constraints, which may require exponential time, and then applying an LP-based solution (one which is complete over the rationals). The crux of the coNP-completeness proof is that we rely on the *generator representation* of the (integer-hull of) the transition polyhedron. We provide sufficient and necessary conditions for the existence of a linear ranking function that use the vertices and rays of this representation. This also leads to an alternative synthesis algorithm.

Another positive aspect of our results, 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.

But linear ranking functions do not suffice for all loops, and, in particular for multipath loops, *lexicographic-linear ranking functions* are a natural extension. Such functions are a tuple of affine functions, such that in every iteration of the loop, the value of the tuple decreases lexicographically. Such a function will work, for example, for the following multipath loop

$$\mathit{loop} : \{x_1 \geq 0, x_2 \geq 0, x'_1 = x_1 - 1\} \vee \{x_1 \geq 0, x_2 \geq 0, x'_2 = x_2 - 1, x'_1 = x_1\} \tag{3}$$

where in the first path x_1 decreases towards zero and x_2 is changed unpredictably, since there is no constraint on x'_2; this could arise, for instance, from x_2 being set to the result of an input from the environment, or a function call for which we have no invariants. In the second path x_2 decreases towards zero and x_1 is unchanged. Clearly, $\langle x_1, x_2 \rangle$ always decreases lexicographically, but there can be no single linear ranking function for this loop.

In Section 5 we analyze the complexity of the decision problem: is there a lexicographic-linear ranking function for a given loop? We also give a complete synthesis algorithm. Our point of departure (corresponding to the case of linear ranking functions) is the known polynomial-time algorithm of Alias et al. [2010], based on LP, that is claimed to be complete—and as explained above, is only complete when one extends the domain of the variables to the rationals. We show that the corresponding decision problem is, like the case of linear ranking function, coNP-complete when the variables are restricted to hold integers. We also give a novel complete synthesis algorithm. The algorithm is of exponential-time complexity, but becomes polynomial-time in special cases corresponding to those identified in the context of linear ranking functions.

We also consider the application of the algorithm to the setting of rational data; in this setting it has polynomial-time complexity and extends the one of Alias et al. [2010], because our class of ranking functions is more general. The algorithm produces a function that descends lexicographically in the rationals; for example, if it produces

$\langle x_1, x_2 \rangle$, it ensures that in every possible transition either $x_1 > x_1'$ and $x_1 \geq 0$ or $x_1 = x_1'$ and $x_2 > x_2'$ and $x_2 \geq 0$. If one is only interested in integer data, such a function proves termination, and this relaxation to the rationals is therefore sound. Over the rationals, however, this lexicographic order is not well-founded — simply because the order $(>)$ on $\mathbb{Q}_+$ is not (consider the sequence $x_1 = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$). Interestingly, we prove that a function that descends in the lexicographic extension of the order $(>)$ can always be turned into one that descends in the lexicographic extension of the order $a \succeq b$ (defined as $b \geq a + 1$), and therefore implies termination.

We prove some properties of our synthesis algorithm, for example that the dimension (the length of the tuple) of the functions it produces is always the smallest possible.

Our results should be of interest to all users of ranking functions, and in fact their use goes 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* [Albert et al. 2011; Alias et al. 2010] and to automatically parallelize computations [Feautrier 1992a; Darte 2010]. We remark that in termination analysis, the distinction between integers and rationals has already been considered, both regarding ranking-function generation [Feautrier 1992a; Bradley et al. 2005c; Cook et al. 2010] and the very decidability of the termination problem [Ben-Amram et al. 2012; Tiwari 2004; Braverman 2006]. 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. [2005c], for synthesizing linear ranking functions, complete (see Section 7).

Our tool iRANKFINDER implements the algorithms mentioned above (and more) and can be tried out online (see Section 6).

This paper is organized as follows. Section 2 gives definitions and background information regarding linear-constraint loops, linear and lexicographic-linear ranking functions, and the mathematical notions involved. Section 3 proves that the decision problem "is there a linear ranking function for an integer loop", is coNP-complete, and also presents an exponential-time ranking-function synthesis algorithm. Section 4 discusses PTIME-solvable cases. Section 5 studies the complexity of the decision problem "is there a lexicographic-linear ranking function for a given loop", both for integer and rational data, and proves that it is coNP-complete and PTIME respectively. It also develops corresponding complete synthesis algorithms. Section 6 describes a prototype implementation. Section 7 surveys related previous work. Section 8 concludes. A conference version of this paper, including the results on linear ranking functions (but not lexicographic-linear ranking functions), has been presented at POPL 2013 [Ben-Amram and Genaim 2013].

2. PRELIMINARIES

In this section we recall some results on (integer) polyhedra on which we will rely along the paper, define the kind of loops we are interested in, and define the *linear* and *lexicographic-linear ranking function* problems for such loops.

2.1. Integer Polyhedra

We recall some useful definitions and properties, all following Schrijver [1986].

Polyhedra. 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 $A\mathbf{x} \leq \mathbf{b}$. We use calligraphic letters, such as $\mathcal{P}$ and $\mathcal{Q}$ to denote polyhedra. The set of *recession directions* of a polyhedron $\mathcal{P}$ specified by $A\mathbf{x} \leq \mathbf{b}$ is the set $\mathcal{R}_{\mathcal{P}} = \{\mathbf{y} \in \mathbb{Q}^n \mid A\mathbf{y} \leq \mathbf{0}\}$.

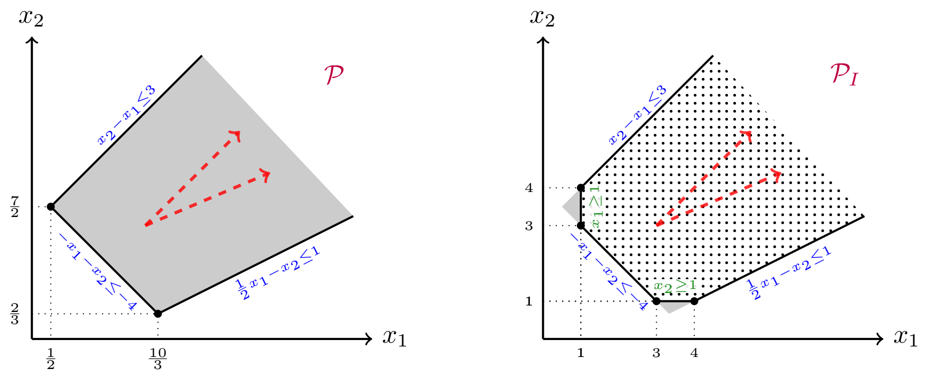

Fig. 1. A polyhedron $\mathcal{P}$ and its integet-hull $\mathcal{P}_I$.

Example 2.1. Consider the polyhedron $\mathcal{P}$ of Figure 1 (on the left). The points defined by the gray area, and the black borders, are solutions to the system of linear inequalities $x_2 - x_1 \leq 3 \wedge -x_1 - x_2 \leq -4 \wedge \frac{1}{2}x_1 - x_2 \leq 1$.

Integer Polyhedra. 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. This property is fundamental to our results. 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*.

Example 2.2. The integer hull $\mathcal{P}_I$ of polyhedron $\mathcal{P}$ of Figure 1 (on the left) is given in the same figure (on the right). It is defined by the dotted area and the black border, and is obtained by adding the inequalities $x_1 \geq 1$ and $x_2 \geq 1$ to $\mathcal{P}$. The two gray triangles next to the edges of $\mathcal{P}_I$ are subsets of $\mathcal{P}$ that were eliminated when computing $\mathcal{P}_I$.

Generator representation. Polyhedra also have a *generator representation* in terms of vertices and rays[1], written as

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

This means that $\mathbf{x} \in \mathcal{P}$ if and only if $\mathbf{x} = \sum_{i=1}^m a_i \cdot \mathbf{x}_i + \sum_{j=1}^t b_j \cdot \mathbf{y}_j$ 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., $\mathbf{y} \in \mathcal{R}_\mathcal{P}$ if and only if $\mathbf{y} = \sum_{j=1}^t b_j \cdot \mathbf{y}_j$ for some rationals $b_j \geq 0$. If $\mathcal{P}$ is integral, then there is a generator representation in which all $\mathbf{x}_i$ and $\mathbf{y}_j$ are integer. An empty polyhedron is represented by an empty set of vertices and rays.

Example 2.3. The generator representations of $\mathcal{P}$ and $\mathcal{P}_I$ of Figure 1 are

$$\begin{aligned} \mathcal{P} &= \text{convhull}\{(\tfrac{1}{2}, \tfrac{7}{2}), (\tfrac{10}{3}, \tfrac{2}{3})\} + \text{cone}\{(1,1), (7,3)\} \\ \mathcal{P}_I &= \text{convhull}\{(1,3), (1,4), (3,1), (4,1)\} + \text{cone}\{(1,1), (7,3)\} \end{aligned}$$

The points in convhull are vertices, they correspond to the points marked with $\bullet$ in Figure 1. The rays are the vectors $(1,1), (7,3)$; they describe a direction, rather than a specific point, and are therefore represented in the figure as arrows. Note that the vertices of $\mathcal{P}_I$ are integer points, while those of $\mathcal{P}$ are not. The point $(3,2)$, for example,

[1]Technically, the $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are only vertices if the polyhedron is *pointed*.

is defined as $\frac{5}{17}\cdot(\frac{1}{2},\frac{7}{2})+\frac{12}{17}\cdot(\frac{10}{3},\frac{2}{3})\}+\frac{1}{2}\cdot(1,1)+0\cdot(7,3)$ in $\mathcal{P}$, and as $0\cdot(1,3)+\frac{1}{3}\cdot(1,4)+0\cdot(3,1)+\frac{2}{3}\cdot(4,1)+0\cdot(1,1)+0\cdot(7,3)$ in $\mathcal{P}_I$.

Faces. If $\mathbf{c}$ is a nonzero vector and $a=\max\{\mathbf{c}\cdot\mathbf{x}\mid\mathbf{x}\in\mathcal{P}\}$, then $\mathcal{H}=\{\mathbf{x}\in\mathbb{Q}^n\mid\mathbf{c}\cdot\mathbf{x}=a\}$ is called a supporting hyperplane for $\mathcal{P}$. A *non-empty* subset $\mathcal{F}\subseteq\mathcal{P}$ is called a *face* if $\mathcal{F}=\mathcal{P}$ or $\mathcal{F}$ is an intersection of $\mathcal{P}$ with a supporting hyperplane [Schrijver 1986, p. 101]. In the latter case we say that $\mathcal{F}$ is a *proper* face of $\mathcal{P}$. Alternatively, $\mathcal{F}$ is face of $\mathcal{P}$ if and only if it can be obtained by turning some inequalities of $A\mathbf{x}\le b$ to equalities [Schrijver 1986, Sec. 16.3, p. 231]. It is known that a polyhedron $\mathcal{P}$ is integral if and only if every face of $\mathcal{P}$ includes an integer point [Schrijver 1986, Sec. 16.3, p. 231]. This implies that the faces of an integral polyhedron $\mathcal{P}$ are integral.

Example 2.4. Polyhedron $\mathcal{P}$ of Figure 1 has 5 *proper* faces, each corresponds to either a black segment or a vertex (a point marked with $\bullet$). For example, the segment between $(\frac{1}{2},\frac{7}{2})$ and $(\frac{10}{3},\frac{2}{3})$ is a proper face, and it can be obtained by turning the inequality $-x_1-x_2\le-4$ to $-x_1-x_2=-4$ in $\mathcal{P}$. Similarly, polyhedron $\mathcal{P}_I$ of Figure 1 has 9 *proper* faces, in this case each includes an integer point.

Dimension of polyhedra. Let $A^{=}\mathbf{x}\le\mathbf{b}^{=}$ be the set of all implicit equalities in $A\mathbf{x}\le\mathbf{b}$ ($\mathbf{a}_i\cdot\mathbf{x}\le\mathbf{b}_i$ is an implicit inequality if $\mathbf{a}_i\cdot\mathbf{x}=\mathbf{b}_i$ holds for any $\mathbf{x}\in\mathcal{P}$). The *affine hull* of $\mathcal{P}$ is defined as $\texttt{aff.hull}(\mathcal{P})=\{\mathbf{x}\in\mathbb{Q}^n\mid A^{=}\mathbf{x}=\mathbf{b}^{=}\}$. The dimension of the affine hull is the dimension of the linear subspace $\{\mathbf{x}\mid A^{=}\mathbf{x}=\mathbf{0}\}$ (i.e, the cardinality of the bases). Alternatively, it is equal to n minus the rank of the matrix $A^{=}$. The *dimension* of a polyhedron $\mathcal{P}\subseteq\mathbb{Q}^n$, denoted by $\dim(\mathcal{P})$, is equal to the dimension of its affine hull. The dimension of the empty polyhedron, by convention, is -1. The dimension of a proper face of $\mathcal{P}$ is at least 1 less than that of $\mathcal{P}$. Note that when $\dim(\mathcal{P})=0$ then $\mathcal{P}$ is a single point.

Example 2.5. Both $\mathcal{P}$ and $\mathcal{P}_I$ of Figure 1 have dimension 2. Their proper faces that are defined by segments (resp. vertices) have dimension 1 (resp. 0).

Relative interior. The *relative interior* of $\mathcal{P}$ is defined as $\texttt{ri}(\mathcal{P})=\{\mathbf{x}\mid\exists\epsilon>0\,.\,B(\mathbf{x},\epsilon)\cap\texttt{aff.hull}(\mathcal{P})\subseteq\mathcal{P}\}$ where $B(\mathbf{x},\epsilon)$ is a ball of radius ϵ centered on $\mathbf{x}$. Intuitively, it is the set of all points which are not on the "edge" of $\mathcal{P}$. Note that $\mathbf{x}\in\texttt{ri}(\mathcal{P})$ if and only if $\mathbf{x}\in\mathcal{P}$ and $\mathbf{x}$ does not belong to any proper face of $\mathcal{P}$. When $\dim(\mathcal{P})=0$, the single point of $\mathcal{P}$ is in the relative interior (since $\mathcal{P}$ does not have any proper face).

Example 2.6. Consider the polyhedra of Figure 1. The relative interior of $\mathcal{P}$ is defined by the gray area, and that of $\mathcal{P}_I$ by the dotted area, i.e., we exclude the points on the black segments of each polyhedron (which are proper faces as explained in Example 2.6).

Size of polyhedra. 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 Schrijver [1986, 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 $\mathbf{a}\cdot\mathbf{x}\le c$ as $1+\|c\|+\|\mathbf{a}\|$. For a polyhedron $\mathcal{P}\subseteq\mathbb{Q}^n$ defined by $A\mathbf{x}\le\mathbf{b}$, we let $\|\mathcal{P}\|_b$ be the bit-size of $A\mathbf{x}\le\mathbf{b}$, 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 $\phi\ge n$ such that $\mathcal{P}$ may be described by *some* $A\mathbf{x}\le\mathbf{b}$ where each inequality in $A\mathbf{x}\le\mathbf{b}$ fits in ϕ bits. Clearly, $\|\mathcal{P}\|_f\le\|\mathcal{P}\|_b$. The *vertex size*, denoted by $\|\mathcal{P}\|_v$, is the smallest number $\psi\ge n$ 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 calculated as above). For integer polyhedra, we restrict the generators to be integer. The following theorems state some relations between the different bit-sizes defined above, they are later used to polynomially bound the bit-size of some set of integer points of $\mathcal{P}_I$. They are from Schrijver [1986] (Th. 10.2, p. 121, and Cor. 17.1a,17.1b, p. 238), who cites Karp and Papadimitriou [1980].

THEOREM 2.7. *Let $\mathcal{P}$ be a rational polyhedron in $\mathbb{Q}^n$; then $\|\mathcal{P}\|_v \le 4n^2\|\mathcal{P}\|_f$ and $\|\mathcal{P}\|_f \le 4n^2\|\mathcal{P}\|_v$.*

THEOREM 2.8. *Let $\mathcal{P}$ be a rational polyhedron in $\mathbb{Q}^n$; then $\|\mathcal{P}_I\|_v \le 6n^3\|\mathcal{P}\|_f$ and $\|\mathcal{P}_I\|_f \le 24n^5\|\mathcal{P}\|_f$.*

2.2. Multipath Linear-Constraint Loops

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

$$\textit{while } (B\mathbf{x} \le \mathbf{b}) \textit{ do } A\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} \le \mathbf{c} \tag{4}$$

where $\mathbf{x} = (x_1, \dots, x_n)^{\mathrm{T}}$ and $\mathbf{x}' = (x'_1, \dots, x'_n)^{\mathrm{T}}$ 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 $B\mathbf{x} \le \mathbf{b}$ 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 x (satisfying the loop condition) there is at most one x′ satisfying the update constraint. The update is called *affine linear* if it can be rewritten as

$$\mathbf{x}' = A'\mathbf{x} + \mathbf{c}' \tag{5}$$

for a matrix A' and vector $\mathbf{c}'$ of appropriate dimensions. We say that the loop is a *rational loop* if x and 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 x satisfies the condition and x and 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 x and its last n components to x′. For ease of notation, we denote $\binom{\mathbf{x}}{\mathbf{x}'}$ by 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 the set of inequalities $A''\mathbf{x}'' \le \mathbf{c}''$ where

$$A'' = \begin{pmatrix} B & 0 \\ \multicolumn{2}{c}{A} \end{pmatrix} \qquad\qquad \mathbf{c}'' = \begin{pmatrix}\mathbf{b}\\ \mathbf{c}\end{pmatrix}$$

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., $m \ge 1$ in the generator representation). The polyhedron defined by the loop condition $B\mathbf{x} \le \mathbf{b}$ will be denoted by $\mathcal{C}$ and referred to as 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):

$$\textit{loop} \bigvee_{i=1}^{k} \left[B_i\mathbf{x} \le \mathbf{b}_i \;\wedge\; A_i\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} \le \mathbf{c}_i \right] \tag{6}$$

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 $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ is specified by $A''_i\mathbf{x}'' \le \mathbf{c}''_i$. The polyhedron defined by the condition $B_i\mathbf{x} \le \mathbf{b}_i$ 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

An affine 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 an affine linear function using the row vector $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$. For a given function ρ, we define the function $\Delta\rho : \mathbb{Q}^{2n} \mapsto \mathbb{Q}$ as $\Delta\rho(\mathbf{x}'') = \rho(\mathbf{x}) - \rho(\mathbf{x}')$. Next we define when an affine linear function is a *linear ranking function* (LRF for short) for a given rational or integer MLC loop.

Definition 2.9. Given a set $T \subseteq \mathbb{Q}^{2n}$, representing transitions, we say that ρ is a LRF for T if the following hold for every $\mathbf{x}'' \in T$:

$$\rho(\mathbf{x}) \geq 0\,, \tag{7}$$

$$\Delta\rho(\mathbf{x}'') \geq 1\,. \tag{8}$$

We say that ρ is a LRF for a rational loop, specified by $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$, when it is a LRF for all of $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ (equivalently, it is a LRF for $\bigcup_{i=1}^{k} \mathcal{Q}_i$). We say that ρ is a LRF for an integer loop, specified by $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ polyhedra, when it is a LRF for all of $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$.

Clearly, the existence of a LRF implies termination of the loop. Note that in (8) we require ρ to decrease at least by 1, whereas in the literature [Podelski and Rybalchenko 2004a] this 1 is sometimes replaced by $\delta > 0$. It is easy to verify that these definitions are equivalent as far as the existence of a LRF is concerned.

Definition 2.10. 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 [Podelski and Rybalchenko 2004a; Mesnard and Serebrenik 2008]. In this paper, we focus on LINRF($\mathbb{Z}$).

2.4. Lexicographic-Linear Ranking Functions

A d*-dimensional affine function* $\tau : \mathbb{Q}^n \to \mathbb{Q}^d$ is a function of the form $\tau = \langle \rho_1, \ldots, \rho_d \rangle$, where each component $\rho_i : \mathbb{Q}^n \to \mathbb{Q}$ is an affine function. The number d is informally called the *dimension* of the function (technically, it is the dimension of the co-domain). Next we define when a d-dimensional affine function is a *lexicographic-linear ranking function* ($LLRF$ for short) for a given rational or integer MLC loop.

Definition 2.11. Let $T \subseteq \mathbb{Q}^{2n}$ be a given set, representing transitions, and $\tau = \langle \rho_1, \ldots, \rho_d \rangle$ a d-dimensional affine function. We say that τ is a $LLRF$ for T if and only if for every $\mathbf{x}'' \in T$ there exists $i \leq d$ such that the following hold

$$\forall j < i\,.\ \Delta\rho_j(\mathbf{x}'') \geq 0\,, \tag{9}$$

$$\forall j \leq i\,.\quad \rho_j(\mathbf{x}) \geq 0\,, \tag{10}$$

$$\Delta\rho_i(\mathbf{x}'') \geq 1\,. \tag{11}$$

We say that $\mathbf{x}''$ is ranked by ρ_i.

As for $LRFs$, we say that τ is a $LLRF$ for a rational loop $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ when it is a $LLRF$ for $\bigcup_{i=1}^{k} \mathcal{Q}_i$, and that it is a $LLRF$ for the corresponding integer loop if it is a $LLRF$ for $\bigcup_{i=1}^{k} I(\mathcal{Q}_i)$.

Note that in (11) we require ρ_i to decrease at least by 1. As for the case of $LRFs$, this 1 can be replaced by any $\delta_i > 0$. It is easy to verify that these definitions are equivalent as far as the existence of a $LLRF$ is concerned. The existence of a $LLRF$ implies termination of the loop. This may be justified by converting the function into one that decreases in a well-founded set; such a function is

$$\hat{\tau}(\mathbf{x}) = \langle \max(0, \rho_1(\mathbf{x})), \ldots, \max(0, \rho_d(\mathbf{x})) \rangle,$$

whose co-domain is $\langle \mathbb{Q}_+^d, \preceq_{lex} \rangle$, where $\preceq_{lex}$ is the lexicographic extension of the well-founded order: $a \preceq b$ iff $a + 1 \leq b$.

Our class of $LLRF$s differs somewhat from other classes of "lexicographic-linear ranking functions" that appeared in the literature [Bradley et al. 2005a; Alias et al. 2010]. Specifically, the definition in Alias et al. [2010] is more restrictive since it requires (10) to hold for all $1 \leq j \leq d$. The following example illustrates the difference.

Example 2.12. Consider the SLC loop

$$\mathit{while}(x_1 \geq 0, x_2 \geq 0, x_3 \geq -x_1)\ \mathit{do}\ x_2' = x_2 - x_1, x_3' = x_3 + x_1 - 2\,. \tag{12}$$

It has a $LLRF$ $\tau = \langle x_2, x_3 \rangle$ as in Definition 2.11 (over both rationals and integers), however, it does not have a $LLRF$ according to Alias et al. [2010]. Indeed, when x_2 decreases x_3 can be negative (e.g., for $x_1 = 1$, $x_2 = 2$ and $x_3 = -1$).

Another difference from Alias et al. [2010] lies in the fact that they require the non-negativity conditions (10) to be implied by *the loop guard*. That is, it is not possible to use the constraints in the update part of the loop in proving this condition, when according to our definition it is possible.

The definition of Bradley et al. [2005a] requires (10) to hold only for $j = i$, which adds flexibility, as we show next.

Example 2.13. Consider the MLC loop

$$\mathit{loop} : \{x_1 \geq 0, x_1' = x_1 - 1\} \vee \{x_2 \geq 0, x_2' = x_2 - 1, x_1' \leq x_1\}\,. \tag{13}$$

It has a $LLRF$ $\tau = \langle x_1, x_2 \rangle$ according to the definition of Bradley et al. [2005a], however, it does not have one that satisfies Definition 2.11. Indeed, in transitions where x_2 decreases x_1 may be negative, but x_1 must be the first component.

Another difference is that Bradley et al. [2005a] require a fixed association of ranking-function components with the paths of the loop. So, for example, they cannot have a 2-dimensional $LLRF$ for an SLC loop, as in Example 2.12.

Definition 2.14. The decision problem *Existence of a* $LLRF$ is defined by

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

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

3. LINRF($\mathbb{Z}$) 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 that restrict the variables to a finite range. We also show that $LRFs$ can be synthesized in deterministic exponential time.

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

3.1. coNP-hardness

We prove coNP-hardness in a strong form. Recall that a number problem (a problem whose instance is a matrix of integers) **Prob** is strongly hard for a complexity class, if there are polynomial reductions from all problems in that class to **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 [Garey and Johnson 1979].

THEOREM 3.1. *The* $\textsc{LinRF}(\mathbb{Z})$ *problem is strongly coNP-hard, even for* SLC *loops with affine-linear updates.*

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 [Karp 1972]). We reduce this problem to $\textsc{LinRF}(\mathbb{Z})$.

Given $B \in \mathbb{Z}^{m\times n}$ and $\mathbf{b} \in \mathbb{Z}^m$, we construct the following integer SLC loop

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

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathsf{T}}$, $\mathbf{z} = (z_1, \ldots, z_m)^{\mathsf{T}}$ 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, it is easy to see that 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 alone 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 alrady 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.

Note also that the loop constructed in the reduction either has a LRF, or fails to terminate. Hence, one cannot hope to avoid the coNP-hardness by using another kind of certificate instead of linear ranking functions, as long as the certificate is sufficiently expressive to capture the termination argument for integer loops where variables are limited to $[0, 1]$, update is an affine linear function, and termination follows from the fact that a sum of variables always descends.

3.2. Inclusion in coNP for SLC Loops

To prove that $\textsc{LinRF}(\mathbb{Z})$ is in coNP, we show that the complement of $\textsc{LinRF}(\mathbb{Z})$, 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 $I(\mathcal{Q})$ if and only if 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 $\mathbf{x}'' \in I(\mathcal{Q})$ is a witness *against* $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ if it fails to satisfy at least one of

$$\vec{\lambda} \cdot \mathbf{x} + \lambda_0 \geq 0 \tag{14}$$

$$\vec{\lambda} \cdot (\mathbf{x} - \mathbf{x}') \geq 1 \tag{15}$$

The set of $(\lambda_0,\vec{\lambda})$ witnessed against by $\mathbf{x}''$ is denoted by $W(\mathbf{x}'')$.

Note that conditions (14,15) are obtained from (7,8) by writing ρ explicitly.

Definition 3.3. We say that $\mathbf{y}'' \in I(\mathcal{R}_{\mathcal{Q}})$ 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

$$\vec{\lambda} \cdot \mathbf{y} \geq 0 \tag{16}$$

$$\vec{\lambda} \cdot (\mathbf{y} - \mathbf{y}') \geq 0 \tag{17}$$

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 Definition 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 (14) for any point of the form $\mathbf{x}'' + a\mathbf{y}''$ with integer $a > 0$ since it is a point in $I(\mathcal{Q})$; letting a grow to infinity, we see that (14) implies the homogeneous inequality (16). Similarly, (15) implies (17).

Definition 3.4. Given $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$, define

$$WS(X,Y) = \bigcup_{\mathbf{x}'' \in X} W(\mathbf{x}'') \cup \bigcup_{\mathbf{y}'' \in Y} W_H(\mathbf{y}'') . \tag{18}$$

LEMMA 3.5. *Let $X \subseteq I(\mathcal{Q})$, $X \neq \emptyset$, 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 one of the conditions(14) and (15) does 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 (16,17). Next we show that, in such case, there must exist $\mathbf{z}'' \in I(\mathcal{Q})$ that fails either (14) or (15). In this part of the proof, we rely on the fact that $X \neq \emptyset$.

Case 1: Suppose (16) is not satisfied. That is, $\vec{\lambda} \cdot \mathbf{y} < 0$.

Choose $\mathbf{x}'' \in X$, and note that $\rho(\mathbf{x}) \geq 0$, otherwise $(\lambda_0,\vec{\lambda}) \in W(\mathbf{x}'')$ which we have assumed not true. 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 $\mathbf{z}'' = \binom{\mathbf{x}+a\cdot\mathbf{y}}{\mathbf{x}'+a\cdot\mathbf{y}'}$. We choose a as an integer sufficiently large so that $a\cdot(\vec{\lambda}\cdot\mathbf{y}) \le -(1+\rho(\mathbf{x}))$. Now,

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

So ρ fails (14) on $\mathbf{z}'' \in I(\mathcal{Q})$, and thus cannot be a LRF.

Case 2: Suppose (17) is not satisfied. That is, $\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}') < 0$.

Choose $\mathbf{x}'' \in X$, and note that $\rho(\mathbf{x})-\rho(\mathbf{x}') \ge 1$, otherwise $(\lambda_0,\vec{\lambda}) \in W(\mathbf{x}'')$ which we have assumed not true. Define $\mathbf{z}''$ as above, but now choosing a sufficiently large to make $a\cdot(\vec{\lambda}\cdot(\mathbf{y}-\mathbf{y}')) \le -(1+\rho(\mathbf{x})-\rho(\mathbf{x}'))$. Now,

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

So ρ fails (15) 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 (14,15), for all $\mathbf{x}'' \in X$, and inequalities (16,17), 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:

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

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

$$\{2\lambda_2+\lambda_0 \ge 0,\ -2\lambda_1+\lambda_2 \ge 1,\ \lambda_1-2\lambda_2 \ge 0,\ 2\lambda_1 \ge 0\} \tag{19}$$

The first two inequalities correspond to applying (14,15) to $\mathbf{x}_1''$, and the other ones to applying (16,17) to $\mathbf{y}_1''$. It is easy to verify that (19) 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 also 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 respectively.

LEMMA 3.7. *Let the integer hull of the transition polyhedron $\mathcal{Q}$ be $\mathcal{Q}_I = \mathrm{convhull}\{\mathbf{x}_1'',\ldots,\mathbf{x}_m''\} + \mathrm{cone}\{\mathbf{y}_1'',\ldots,\mathbf{y}_t''\}$. 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

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

Then, there is $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ that fulfills (14,15) for all $\mathbf{x}_i''$ and (16,17) 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 $\mathbf{x}''$ 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 (14,15) which means that ρ is a LRF for $I(\mathcal{Q})$:

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

□

Note that the solutions of $\Psi_{WS}(\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\})$ in Lemma 3.7 actually define the set of all $LRFs$ for $I(\mathcal{Q})$. We will address this point later in Section 3.4, for synthesizing $LRFs$.

Example 3.8. Consider again the loop of Example 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

$$\left\{\begin{matrix} \lambda_2 + \lambda_0 \geq 0, & -\lambda_2 \geq 0, & \lambda_2 \geq 0, & \lambda_1 - \lambda_2 \geq 0, \\ -\lambda_1 + \lambda_2 \geq 1, & \lambda_1 \geq 0, & -\lambda_1 \geq 0, & \lambda_1 \geq 0 \end{matrix}\right\} \tag{20}$$

The inequalities in the leftmost column correspond to applying (14,15) to $\mathbf{x}''_1$, and those in the other columns to applying (16,17) to $\mathbf{y}''_1$, $\mathbf{y}''_2$, and $\mathbf{y}''_3$ respectively. It is easy to verify that (20) is not satisfiable, and thus, $WS(\{\mathbf{x}''_1\},\{\mathbf{y}''_1, \mathbf{y}''_2, \mathbf{y}''_3\}) = \mathbb{Q}^3$.

Lemmas 3.5 and 3.7 provide a necessary and sufficient condition for the nonexistence of a LRF.

COROLLARY 3.9. *There is no LRF for $I(\mathcal{Q})$ if and only if there are two finite sets $X \subseteq I(\mathcal{Q})$, $X \neq \emptyset$, 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 a LRF for $I(\mathcal{Q})$, there exists one with $X \subseteq I(\mathcal{Q})$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}})$ such that $|X| + |Y| \leq n+2$; and its bit-size is polynomially bounded 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 [Schrijver 1986, p. 94] states that if a finite set of inequalities over $\mathbb{Q}^d$, for some $d > 0$, has no solution, there is a subset of at most $d+1$ 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 $n+2$ inequalities that has no solution. This subset involves at most $n+2$ integer points out of $\{\mathbf{x}''_1, \ldots, \mathbf{x}''_m\}$ and $\{\mathbf{y}''_1, \ldots, \mathbf{y}''_t\}$, because every inequality in $\Psi_{WS}(\{\mathbf{x}''_1, \ldots, \mathbf{x}''_m\},\{\mathbf{y}''_1, \ldots, \mathbf{y}''_t\})$ is defined by either one $\mathbf{x}''_i$ or $\mathbf{y}''_i$ (see (14–17)). Let these points be X and Y, then $|X|+|Y| \leq n+2$ and $\Psi_{WS}(X,Y)$ has no solution, i.e., $WS(X,Y) = \mathbb{Q}^{n+1}$. Moreover it must be that case that $X \neq \emptyset$, since all constraints of the type (16,17) are satisfied by $(\lambda_0,\vec{\lambda}) = \mathbf{0}^{\mathrm{T}}$.

Now we show that X and Y may be chosen to have bit-size polynomial in the size of the input. Recall that the input is the set of inequalities $A''\mathbf{x}'' \leq \mathbf{c}''$ that define $\mathcal{Q}$, and its bit-size is $\|\mathcal{Q}\|_b$. Recall that the points of X and 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 and Y may be bounded by $(n+2)\cdot\|\mathcal{Q}_I\|_v$. By Theorem 2.8, since the dimension of $\mathcal{Q}$ is $2n$,

$$(n+2)\cdot\|\mathcal{Q}_I\|_v \leq (n+2)\cdot(6\cdot(2n)^3\cdot\|\mathcal{Q}\|_f) \leq (48n^4+96n^3)\cdot\|\mathcal{Q}\|_b$$

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 Example 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, these two points witness the nonexistence of a *LRF* (note that this witness consists, in this example, of less than $n+2$ points).

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

PROOF. We show that the complement of LINRF($\mathbb{Z}$) has a polynomially checkable witness. The witness is a listing of sets X and Y of at most $n+2$ 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 $A''\mathbf{y}'' \leq \mathbf{0}$. 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 $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ 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 may come from different transition polyhedra. Namely, $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$ where each $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$. Lemmas 3.5, 3.7, and 3.10, Corollary 3.9, and Theorem 3.12, are rewritten in terms of such witnesses as follows (the proofs are the same unless stated otherwise).

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

Note that $WS(X,Y) = \bigcup_{i=1}^{k} WS(X_i,Y_i)$ and that in the proof of Lemma 3.5 (when re-used to obtain the above lemma) it is necessary to use the condition $Y_i \neq \emptyset \Rightarrow X_i \neq \emptyset$.

LEMMA 3.14. *For $1 \le i \le k$, let $\mathcal{Q}_{iI} = \text{convhull}\{X_i\} + \text{cone}\{Y_i\}$ be the integer hull of $\mathcal{Q}_i$, and define $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$. If there is no LRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, 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 all $I(\mathcal{Q}_i)$. 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 $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, if and only if there are two finite sets $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$, where $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$, and $Y_i \neq \emptyset \Rightarrow X_i \neq \emptyset$, such that $WS(X,Y) = \mathbb{Q}^{n+1}$.*

LEMMA 3.16. *If there exists a witness for the nonexistence of a LRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, then there exists one with $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$, where $X_i \subseteq I(\mathcal{Q}_i)$ and $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$, such that $\sum_{i=1}^{k}(|X_i| + |Y_i|) \le 2n+3$; and its bit-size is polynomially bounded in the bit-size of the input.*

PROOF. Let $\hat{X}_i, \hat{Y}_i$ be the generators of $\mathcal{Q}_{iI}$. First, as in Lemma 3.10, we argue that there is a set of at most $n+2$ inequalities out of $\Psi_{WS}(\bigcup \hat{X}_i, \bigcup \hat{Y}_i)$ that have no solution. These inequalities correspond to $n+2$ points out of the sets $\hat{X}_i, \hat{Y}_i$. Let X_i (respectively Y_i) be the set of points that come from $\hat{X}_i$ (respectively $\hat{Y}_i$). Since $(\lambda_0,\vec{\lambda}) = \mathbf{0}^{\mathsf{T}}$ is not a solution, at least one of the points must come from a set $\hat{X}_i$. But $n+1$ other points might come from sets $\hat{Y}_i$. Since a witness must satisfy $Y_i \neq \emptyset \Rightarrow X_i \neq \emptyset$, we may have to add $n+1$ points to form a valid witness, for a total of $2n+3$ (clearly, $n+1$ can be replaced by k when $k < n+1$). Bounding the bit-size of the witness is done as in Lemma 3.10, but using the $2n+3$ instead of $n+2$, and $\max_i \|\mathcal{Q}_i\|_b$ instead of $\|\mathcal{Q}\|_b$. □

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

PROOF. Almost identical to the proof of Theorem 3.12. Note that the witness is given as $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$, and the verifier should use the appropriate set of constraints 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 again the integer MLC loop (3) from Section 1. It is a classical MLC loop for which there is no LRF. The integer hulls of the corresponding transition polyhedra are

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

where

$$\begin{array}{llll} \mathbf{x}''_1 = (0,0,-1,0)^{\mathsf{T}} & \mathbf{y}''_1 = (0,0,0,-1)^{\mathsf{T}} & \mathbf{y}''_3 = (0,1,0,0)^{\mathsf{T}} & \mathbf{y}''_5 = (0,1,0,1)^{\mathsf{T}} \\ \mathbf{x}''_2 = (0,0,0,-1)^{\mathsf{T}} & \mathbf{y}''_2 = (0,0,0,1)^{\mathsf{T}} & \mathbf{y}''_4 = (1,0,1,0)^{\mathsf{T}} & \mathbf{y}''_6 = (1,0,1,0)^{\mathsf{T}} \end{array}$$

Let us first consider each path separately. We get

$$\Psi_{WS}(\{\mathbf{x}''_1\},\{\mathbf{y}''_1, \mathbf{y}''_2, \mathbf{y}''_3\}) = \{\lambda_0 \ge 0,\ \lambda_1 \ge 1,\ \lambda_2 \ge 0,\ -\lambda_2 \ge 0\} \tag{21}$$

$$\Psi_{WS}(\{\mathbf{x}''_2\},\{\mathbf{y}''_4, \mathbf{y}''_5, \mathbf{y}''_6\}) = \{\lambda_0 \ge 0,\ \lambda_1 \ge 0,\ \lambda_2 \ge 1\} \tag{22}$$

Both (21) and (22) are satisfiable. In fact, their solutions define the corresponding $LRFs$ for each path when considered separately. For the MLC loop, we have that $\Psi_{WS}(\{\mathbf{x}''_1, \mathbf{x}''_2\},\{\mathbf{y}''_1, \ldots, \mathbf{y}''_6\})$ is the conjunction of the inequalities in (21) and (22), 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 (21,22), thus, a possible witness is $X_1 = \{\mathbf{x}_1''\}$, $Y_1 = \{\mathbf{y}_2''\}$, $X_2 = \{\mathbf{x}_2''\}$, $Y_2 = \emptyset$. Note that it consists of less than $2n+3$ points (as $n=2$).

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 ranking function 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 result is directly implied by lemmas 3.13 and 3.14.

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

The following algorithm follows: (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 Section 1. The integer hull of the transition polyhedron is

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

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

$$\{\lambda_1 + \lambda_2 + \lambda_0 \geq 0,\ \lambda_1 + \lambda_0 \geq 0,\ \lambda_1 + \lambda_2 \geq 0,\ \lambda_1 - \lambda_2 \geq 0,\ \lambda_2 \geq 1\} \tag{23}$$

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*. Note that the loop does not terminate when the variables range over $\mathbb{Q}$, e.g., for $x_1 = x_2 = \frac{1}{2}$ (see Figure 2(A)).

Example 3.21. Let us consider now Loop (2) from Section 1. The integer hull of the transition polyhedron is

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

where

$$\begin{array}{llll} \mathbf{x}_1'' = (4,16,1,16)^{\mathrm{T}} & \mathbf{x}_3'' = (2,8,1,8)^{\mathrm{T}} & \mathbf{x}_5'' = (4,1,1,1)^{\mathrm{T}} & \mathbf{y}_1'' = (5,0,2,0)^{\mathrm{T}} \\ \mathbf{x}_2'' = (1,4,0,4)^{\mathrm{T}} & \mathbf{x}_4'' = (1,1,0,1)^{\mathrm{T}} & \mathbf{x}_6'' = (2,1,1,1)^{\mathrm{T}} & \mathbf{y}_2'' = (5,20,2,20)^{\mathrm{T}} \end{array}$$

The formula $\Psi_{WS}(\{\mathbf{x}_1'', \dots, \mathbf{x}_6''\}, \{\mathbf{y}_1'', \mathbf{y}_2''\})$ is the conjunction of the following inequalities (we eliminated clearly redundant ones)

$$\left\{ \begin{array}{llll} \lambda_1 \geq 1, & 4\lambda_1 + \lambda_2 + \lambda_0 \geq 0, & 4\lambda_1 + 16\lambda_2 + \lambda_0 \geq 0, & 2\lambda_1 + \lambda_2 + \lambda_0 \geq 0, \\ 5\lambda_1 + 20\lambda_2 \geq 0, & 2\lambda_1 + 8\lambda_2 + \lambda_0 \geq 0, & \lambda_1 + 4\lambda_2 + \lambda_0 \geq 0, & \lambda_1 + \lambda_2 + \lambda_0 \geq 0 \end{array} \right\} \tag{24}$$

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*. Note 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 Figure 2(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 (23) and (24). 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 $A_i''\mathbf{x} \le \mathbf{c}_i''$ may require exponential time—the number of generators itself may be exponential. Their bit-size, on the other hand, is polynomial by Theorem 2.8. This is interesting, since it yields:

COROLLARY 3.22. *Consider an MLC loop specified by the transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ is specified by $A_i''\mathbf{x} \le \mathbf{c}_i''$. If there is a LRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, there is one whose bit-size is polynomial in the bit-size of $\{A_i''\mathbf{x} \le \mathbf{c}_i''\}$, 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 $\|\mathcal{Q}_{iI}\|_v \le 6(2n)^3 \cdot \|\mathcal{Q}_i\|_f \le 48n^3 \cdot \|\mathcal{Q}_i\|_b$ for an appropriate i. This means that the bit-size of each equation in $\Psi_{WS}(X,Y)$, having one of the forms (14), (15), (16), or (17) is at most[2] $5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b)$. Let $\mathcal{P}$ be the polyhedron defined by $\Psi_{WS}(X,Y)$, then $\|\mathcal{P}\|_f \le 5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b)$. If $\Psi_{WS}(X,Y)$ has a solution, then any vertex of $\mathcal{P}$ is such a solution, and yields a LRF. Using Theorem 2.7, 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:

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

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

We conclude this section by noting that Theorem 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 definitions 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 the *double description method*, such as PPL [Bagnara et al. 2008b], in which both the generators and constraint representations are kept at the same time, can use the algorithm of Theorem 3.19 judiciously when it seems better than algorithms that use the constraints representation [Podelski and Rybalchenko 2004a; Mesnard and Serebrenik 2008].

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 stated by Cook et al. [2010, Lemma 3]).

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

PROOF. Let $\mathcal{Q}$ be an integer polyhedron. ($\Rightarrow$) Suppose that ρ is a LRF for $\mathcal{Q}$, then clearly 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 (7,8) of Definition 2.9 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 (7,8) for any rational

[2]According to Section 2.1, the bit-size of inequality (14) is $\|\vec{\lambda} \cdot \mathbf{x} + \lambda_0 \ge 0\|_b = 1 + \|(1,\mathbf{x})\|_b + \|0\|_b = 5 + \|\mathbf{x}\|_b \le 5 + \|\mathcal{Q}_{iI}\|_v \le 5 + 48n^3 \cdot (\max_i \|\mathcal{Q}_i\|_b)$. The bit-size of (15)-(17) is similar.

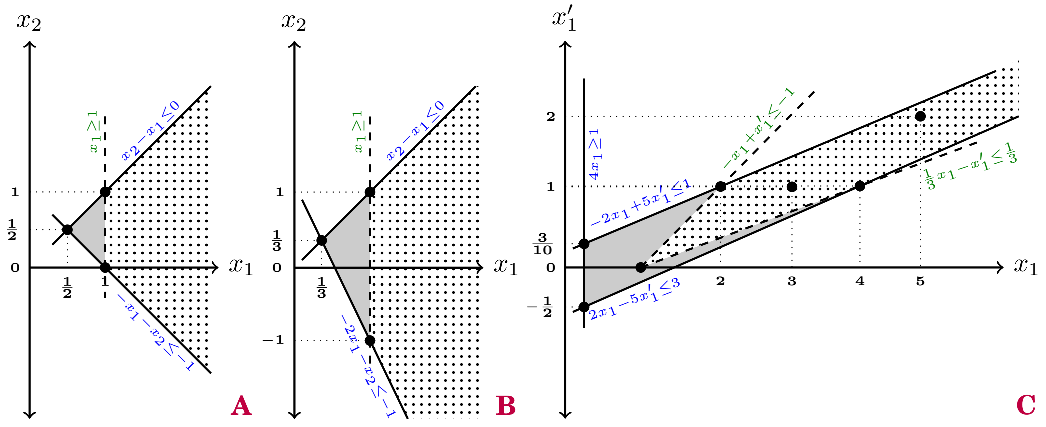

Fig. 2. The polyhedra associated with three of our examples, projected to two dimensions: **(A)** corresponds to Loop (1) at Page 1; **(B)** corresponds to the loop in Example 4.13 at Page 21; and **(C)** corresponds Loop (2) at Page 2. Dashed lines are added when computing the integer hull; dotted areas represent the integer hull; gray areas are rational points eliminated when computing the integer hull.

point in $\mathcal{Q}$, as follows. Choose an arbitrary rational point $\mathbf{x}'' \in \mathcal{Q}$. It can be written as $\mathbf{x}'' = \sum a_i \cdot \mathbf{x}_i''$ where $a_i > 0$, $\sum a_i = 1$ and $\mathbf{x}_i'' \in I(\mathcal{Q})$. Thus, $\mathbf{x}'' = \left(\begin{smallmatrix}\sum a_i \cdot \mathbf{x}_i \\ \sum a_i \cdot \mathbf{x}_i'\end{smallmatrix}\right)$, and

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

□

The above lemma provides an alternative, and *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 larger 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.

Example 4.2. Consider again the SLC loop (2) of Section 1. The transition polyhedron is not integral, computing its integer hull adds the inequalities $-x_1 + x_1' \leq -1$ and $\frac{1}{3}x_1 - x_1' \leq \frac{1}{3}$. This is depicted in Figure 2(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 affine linear with integer coefficients.*

PROOF. We show that if $\mathcal{C}$ is integral, the transition polyhedron $\mathcal{Q}$ is also integral, and thus Corollary 4.3 applies. Let the condition polyhedron $\mathcal{C}$ be integral, and the update be $\mathbf{x}' = A'\mathbf{x} + \mathbf{c}'$ where the entries of A' and $\mathbf{c}'$ are integer. Let $\mathbf{x}'' \in \mathcal{Q}$, 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 $a_i > 0$, $\sum a_i = 1$ 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 $\left(\begin{smallmatrix} \mathbf{x}_i \\ A'\mathbf{x}_i + \mathbf{c}' \end{smallmatrix}\right)$ 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. $\square$

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: Section 4.1 discusses special cases in which the transition or condition polyhedron is integral by construction; Section 4.2 considers cases in which the the transition or condition polyhedron can be separated into independent groups of constraints, each involving few variables; Section 4.3 discusses the case of octagonal relations; Section 4.4 shows that for some cases LINRF$(\mathbb{Z})$ is even strongly polynomial; and Section 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 follows Schrijver [1986], where the proofs of the lemmas can be found.

LEMMA 4.5 ([Schrijver 1986, Eq. (9), p. 230]). *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*

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

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 ([Schrijver 1986, Th. 19.1, p. 266]). *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 affine linear with integer coefficients.*

As a notable example, difference bound constraints [Ben-Amram 2008; Bozzelli and Pinchinat 2012; Bozga et al. 2012] 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 [Schrijver 1986, Th. 20.3, p. 290].

On the other hand, highlighting the gap between linear-ranking proofs and termination proofs in general, we may note that MLC loops with difference bounds, even restricted to the forms $x_i \geq 0$ and $x'_i \leq x_j + c$, already have an undecidable termination problem [Ben-Amram 2008].

4.2. Bounded Number of Variables

In this section we consider cases in which the input loop can be decomposed into different components that do not share variables, and each involves at most N variables for an arbitrary *fixed* N. We start with $N = 2$, and towards the end of this section we consider larger values of N.

Two variable per inequality constraints ($TVPI$ for short) are 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, Figure 2(B). Harvey [1999] 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 constraints. We say that the polyhedron specified by T is a *product of independent two-dimensional $TVPI$ polyhedra* ($PTVPI$ for short), if T 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 $PTVPI$ polyhedra can be computed in polynomial time.*

PROOF. Recall that a polyhedron $\mathcal{P}$ is integral if and only if 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). First we claim that if T_1 and T_2 are two sets of inequalities that do not share variables, and the corresponding polyhedra $\mathcal{T}_1, \mathcal{T}_2$ are integral, then $T_1 \cup T_2$ specifies an integral polyhedron $\mathcal{T}$ over the combined set of variables. Note that $\mathcal{T} = \mathcal{T}_1 \times \mathcal{T}_2$. To prove our claim, note that a face of $\mathcal{T}$ is specified by some constraints defining a face of $\mathcal{T}_1$ and some constraints defining a face of $\mathcal{T}_2$. Since each has an integer point, we get an integer point (in the combined set of variables) satisfying all constraints, i.e., belonging to a face of $\mathcal{T}$.

To compute the integer hull of a $PTVPI$ polyhedron $\mathcal{T}$, we partition its constraints T into independent sets $T_1, \ldots, T_n$, and compute the integer hull of each $\mathcal{T}_i$ in polynomial time using Harvey's method. The above argument shows that $\mathcal{T}_{1I} \times \cdots \times \mathcal{T}_{nI}$ is integral. Moreover, every integer point of $\mathcal{T}$, when projected into the set of variables associated with T_i, is still integer, hence in $\mathcal{T}_{iI}$, which shows that $\mathcal{T}_{1I} \times \cdots \times \mathcal{T}_{nI}$ is the integer hull of $\mathcal{T}$. $\square$

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 any 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 affine linear with integer coefficients.*

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

$$\begin{array}{l} \textit{while } (4x_1 \geq 1,\;\; x_2 \geq 1) \textit{ do} \\ \quad 2x_1 - 5x'_1 \leq 3, \quad -2x_1 + 5x'_1 \leq 1, \;\; x'_2 = x_2 + 1 \end{array} \tag{25}$$

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 the constraints 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 $\mathcal{T}_2$ is already integral. Computing the integer hull of $\mathcal{T}_1$ adds the inequalities $-x_1 + x_1' \leq -1$ and $\frac{1}{3}x_1 - x_1' \leq \frac{1}{3}$. See Figure 2(C). Now LinRF($\mathbb{Q}$) finds the *LRF* $f(x_1, x_2) = x_1 - 1$.

Example 4.13. Consider the following *SLC* loop, as an example for case (2) of Corollary 4.11

$$\begin{array}{l} \textit{while } (-x_1 + x_2 \leq 0, \quad -2x_1 - x_2 \leq -1, \ \ x_3 \leq 1) \textit{ do} \\ \quad x_1' = x_1, \ \ x_2' = x_2 - 2x_1 + x_3, \ \ x_3' = x_3 \end{array} \tag{26}$$

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 $T_1 = \{-x_1 + x_2 \leq 0, \quad -2x_1 - x_2 \leq -1\}$ and $T_2 = \{x_3 \leq 1\}$. It is easily seen that $\mathcal{T}_2$ is already integral; computing the integer hull of $\mathcal{T}_1$ adds $x_1 \geq 1$. See Figure 2(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*.

The special case described above is based on the fact that LinRF($\mathbb{Z}$) for two-dimensional polyhedra is PTIME. In the rest of this section we show that it is PTIME for N-dimensional polyhedra, for a *fixed* constant N, as well. Given a polyhedron $\mathcal{P}$, as a set of linear inequalities $A\mathbf{x} \leq \mathbf{b}$ with n variables and m inequalities, Hartmann [1988, Sec. 4.2] describes an algorithm for computing the vertices $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ of $\mathcal{P}_I$. This algorithm is exponential in the the number of variables n (for fixed n, it is polynomial in the bit-size of $\mathcal{P}$). This means that if we require $n \leq N$, for an arbitrary *fixed* N, we get a polynomial-time algorithm. Note that in such case the number of vertices, ℓ, and the bit-size of each one, are both polynomial in the bit-size of $\mathcal{P}$.

Assuming that $\mathcal{P}$ represents a transition or condition polyhedron, in order to apply LinRF($\mathbb{Q}$) it is not enough to have the vertices of $\mathcal{P}_I$, what we need is a complete representation of $\mathcal{P}_I$ by constraints or by generators . The latter is excluded since the recession cone of $\mathcal{P}_I$ (which is the same as the one of $\mathcal{P}$) can have an exponential number of generators. We next explain how to make use of the constraints representation.

First note that $\mathcal{P}_I = \text{convhull}\{\mathbf{v}_1, \ldots, \mathbf{v}_\ell\} + \mathcal{R}_\mathcal{P}$, where $\mathcal{R}_\mathcal{P}$ is the recession cone of $\mathcal{P}$, and recall that $\mathcal{R}_\mathcal{P} = \{\mathbf{y} \in \mathbb{Q}^n \mid A\mathbf{y} \leq \mathbf{0}\}$. Define the polyhedron $\mathcal{P}'$ as:

$$\mathcal{P}' = \left\{ (\mathbf{x}, \mathbf{a}, \mathbf{y}) \;\middle|\; a_1 \geq 0 \wedge \cdots \wedge a_\ell \geq 0 \wedge \sum_{i=1}^{\ell} a_i = 1 \wedge \mathbf{x} = \left(\sum_{i=1}^{\ell} a_i \mathbf{v}_i \right) + \mathbf{y} \wedge A\mathbf{y} \leq \mathbf{0} \right\}$$

It is easy to see that $\mathbf{x} \in \mathcal{P}_I$ if and only if $(\mathbf{x}, \mathbf{a}, \mathbf{y}) \in \mathcal{P}'$ for some $\mathbf{a}$ and $\mathbf{y}$. The constraint representation for $\mathcal{P}_I$ can be computed by projecting $\mathcal{P}'$ on its first n components $\mathbf{x}$, however, this may take an exponential time. The projection can be avoided by directly using $\mathcal{P}'$, and constraining the *LRF* to not use variables from $(\mathbf{a}, \mathbf{y})$[3]. This yields a polynomial-time algorithm for LinRF($\mathbb{Z}$), for the case of N-dimensional polyhedra, since the bit-size of $\mathcal{P}'$ is polynomial in the bit-size of $\mathcal{P}$.

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*

[3]Such a constraint can be easily imposed when using the Podelski-Rybalchenko procedure, as described in Sections 4.4 and 4.5.

relations [Miné 2006]. 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. For example: 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 [Harvey and Stuckey 1997; Bagnara et al. 2008a; Revesz 2009]. 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 SLC loop

$$\begin{array}{l} \mathit{while}\ (x_1 + x_2 \leq 2,\ \ x_1 + x_3 \leq 3,\ \ x_2 + x_3 \leq 4)\ \mathit{do} \\ \quad x_1' = 1 - x_1,\ \ x_2' = 1 + x_1,\ \ x_3' = 1 + x_2 \end{array} \tag{27}$$

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 Loop (1) of Section 1 in which the condition is an octagonal relation. LINRF($\mathbb{Q}$) fails to find a LRF since the loop may fail to terminate for rational-valued variables. Computing the tight closure of the condition polyhedron adds the inequality $x_1 \geq 1$, making the polyhedron integral. See Figure 2(A). Now LINRF($\mathbb{Q}$) finds the LRF $f(x_1, x_2) = x_1 + x_2 - 1$. Let us consider an example with higher dimensions

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

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 $-x_3 \leq -1$, 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 [Schrijver 1986, 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 affine 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 Section 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 affine linear with integer coefficients—so at least for this class there is, presumably, no polynomial solution. We present this reduction next.

THEOREM 4.17. *The* LINRF($\mathbb{Z}$) *problem is strongly coNP-hard, even for deterministic SLC loops where the guard is octagonal.*

PROOF. We exhibit a polynomial-time reduction from 3SAT to the complement of LINRF($\mathbb{Z}$) (keeping all the numbers in the resulting instance polynomially bounded, to obtain strong coNP-hardness).

Consider a 3SAT instance given as a collection of m clauses, $C_1, \dots, C_m$, each clause C_i consisting of three literals $L_i^j \in \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$. We construct a loop over $4m$ variables. Variable x_{ij} corresponds to L_i^j. Variable x_{i0} is a control variable to ensure the satisfaction of clause i, as will be seen below. Let C be the set of all conflicting pairs, that is, pairs $((i,j),(r,s))$ such that L_i^j is the complement of L_r^s, and also pairs $((i,j),(i,j'))$ with $1 \leq j < j' \leq 3$. The loop we construct is:

$$\begin{aligned}
\textit{while} & \left(\bigwedge_{((i,j),(r,s))\in C} x_{ij} + x_{rs} \leq 1 \right) \wedge \left(\bigwedge_{1\leq i\leq m,\ 0\leq j\leq 3} 0 \leq x_{ij} \leq 1 \right) \\
\textit{do} & \left(\bigwedge_{1\leq i\leq m,\ 1\leq j\leq 3} x'_{ij} = x_{ij} \right) \wedge \left(\bigwedge_{1\leq i\leq m} x'_{i0} = x_{i0} + x_{i1} + x_{i2} + x_{i3} - 1 \right)
\end{aligned}$$

Suppose the formula is satisfiable. For every clause, choose a satisfied literal, and set the corresponding variable x_{ij} to 1; let all other variables be zero. Observe that all the inequality constraints are fulfilled, and that the value of each x'_{i0} does not change. Hence, the loop does not terminate, and does not have any ranking function, let alone a *LRF*.

Next, suppose the formula is unsatisfiable. An initial state for which the loop guard is enabled may be interpreted as a selection of non-conflicting literals. Since no such selection can satisfy all clauses, looking at the update of the x_{i0} variables, we see that some may stay unchanged, while some (and at least one) will decrease. It follows that $\sum_i x_{i0}$ is a LRF. $\square$

4.4. Strongly Polynomial Cases

Polynomial-time algorithms for LINRF($\mathbb{Q}$) [Podelski and Rybalchenko 2004a; Mesnard and Serebrenik 2008; Alias et al. 2010] 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 [1986] (quoting Schrijver [1986, p. 196]).

THEOREM 4.18 (TARDOS). *There is an algorithm which solves a rational LP problem $\max\{\mathbf{c}\cdot\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ 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 4.19. *There exists a strongly polynomial algorithm to solve an LP problem $\max\{\mathbf{c}\cdot\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ 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 [Podelski and Rybalchenko 2004a], slightly modified to require that the LRF decreases at least by 1 instead of by some $\delta > 0$ (this modification only affects (28e) below; the right-hand side of the inequality is $-\delta$, so in their formulation the inequality was $\vec{\eta}\cdot\mathbf{c}'' < 0$).

THEOREM 4.20 (PODELSKI-RYBALCHENKO). *Given an SLC loop with a transition polyhedron $\mathcal{Q} \subseteq \mathbb{Q}^{2n}$, specified by $A''\mathbf{x}'' \leq \mathbf{c}''$, let $A'' = (A\ A')$ where each A and A' has n columns and m rows each, and let $\vec{\mu}, \vec{\eta}$ be row vectors of different m rational variables each. A LRF for $\mathcal{Q}$ exists if and only if there is a (rational) solution to the following set of constraints*

$$\vec{\mu}, \vec{\eta} \geq \mathbf{0}^T, \tag{28a}$$

$$\vec{\mu}\cdot A' = \mathbf{0}^T, \tag{28b}$$

$$(\vec{\mu} - \vec{\eta})\cdot A = \mathbf{0}^T, \tag{28c}$$

$$\vec{\eta}\cdot(A + A') = \mathbf{0}^T, \tag{28d}$$

$$\vec{\eta}\cdot\mathbf{c}'' \leq -1\,. \tag{28e}$$

THEOREM 4.21. *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 $\{0, \pm 1\}$.*

PROOF. First observe that, in Theroem 4.20, when the matrix A'' has only entries from $\{0, \pm 1\}$, then all coefficients in the constraints (28a–28d) are from $\{0, \pm 1, \pm 2\}$. Moreover, the number of inequalities and variables in (28a–28d) 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 (28a–28e), we consider the minimization of $\vec{\eta} \cdot \mathbf{c}''$ under the other constraints (28a–28d). Clearly, this answers the same question since: (28a–28e) is feasible, if and only if the minimization problem is unbounded, or the minimum is negative. This brings the problem to the form required by Corollary 4.19 and yields our result. □

COROLLARY 4.22. *The* LINRF($\mathbb{Z}$) *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 $\{0, \pm 1\}$, that are covered by any of the special cases of Section 4.1 and the special case of $PTVPI$ constraints of Section 4.2.*

PROOF. In the cases of Section 4.1, the transition polyhedron is guaranteed to be integral. In the $PTVPI$ case of Section 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 $\{0, \pm 1\}$. 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 $\{0, \pm 1\}$, and the number of such constraints is polynomially bounded by the number of the original inequalities. Thus, by Theorem 4.21, we can apply a strongly polynomial-time algorithm for LINRF($\mathbb{Q}$). □

4.5. Multipath Loops

It follows immediately from the definitions that an affine linear function ρ is a LRF for an MLC loop with transition polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$ if and only if 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 (28a–28e) defines the set of $LRFs$ for the corresponding SLC loop as follows.

LEMMA 4.23. *Given an SLC loop with a transition polyhedron $\mathcal{Q}$, specified by $A''\mathbf{x}'' \le \mathbf{c}''$, let $\Gamma(\vec{\mu}, \vec{\eta}, A'', c'')$ be the conjunction of (28a–28e). Then, $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$ is a LRF for $\mathcal{Q}$ if and only if $\Gamma(\vec{\mu}, \vec{\eta}, A'', c'')$ has a solution such that $\vec{\lambda} = \vec{\eta} \cdot A'$ and $\lambda_0 \ge \vec{\mu} \cdot \mathbf{c}''$.*

Next we show 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 stated by Cook et al. [2010, Lemma 3]).

THEOREM 4.24. *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 $\Gamma(\vec{\mu}_i, \vec{\eta}_i, A_i'', c_i'')$ be the constraints (28a–28e) for the i-th path, and $(\lambda_0, \vec{\lambda})$ be $n+1$ rational variables. Then there is a LRF for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$ if and only if the following is feasible (over the rationals)*

$$\bigwedge_{i=1}^{k} \Gamma(\vec{\mu}_i, \vec{\eta}_i, A_i'', c_i'') \wedge \vec{\lambda} = \vec{\eta}_i \cdot A_i' \wedge \lambda_0 \ge \vec{\mu}_i \cdot \mathbf{c}_i'' \tag{29}$$

Moreover, the values of $(\lambda_0, \vec{\lambda})$ in the solutions of (29) *define the set of all $LRFs$ for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$.*

PROOF. Immediate by Lemma 4.23, noting that for each $1 \leq i \leq k$ the constraints $\Gamma(\vec{\mu}_i, \vec{\eta}_i, A''_i, c''_i)$ uses different $\vec{\mu}_i$ and $\vec{\eta}_i$, while $(\lambda_0,\vec{\lambda})$ are the same for all i. □

COROLLARY 4.25. *The* LINRF($\mathbb{Q}$) *problem for MLC loops is PTIME-decidable.*

PROOF. The size of the set of inequalities (29) 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 4.26. *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 sections 4.1 and 4.2.*

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

Example 4.27. Consider an MLC loop with the following two paths: Loop (1) of Section 1; and the loop of Example 4.12. Applying LINRF($\mathbb{Q}$) (as in Theorem 4.24) 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 affine linear with integer coefficients. That of the second path has been computed in Example 4.12.

We conclude our discussion on the special PTIME cases for LINRF($\mathbb{Z}$) with a summary table (Figure 3), that briefly describes each case and illustrates it with an example.

5. THE LEXICOGRAPHIC-LINEAR RANKING PROBLEM

In this section we turn to the problems of finding a Lexicographic-Linear Ranking Function ($LLRF$), or determining if one exist (as defined in Section 2.4). We study the complexity of both LEXLINRF($\mathbb{Z}$) and LEXLINRF($\mathbb{Q}$) and develop corresponding complete algorithms for synthesizing $LLRFs$ (moreover, $LLRFs$ of smallest dimension).

In Section 5.1 we consider the LEXLINRF($\mathbb{Z}$) problem, and develop a synthesis algorithm which has exponential-time complexity in general, and polynomial-time complexity for the special cases of Section 4. We also provide sufficient and necessary conditions for the existence of a $LLRF$ which imply the completeness of our algorithm. These conditions are used in Section 5.2 to show that LEXLINRF($\mathbb{Z}$) is coNP-complete.

In Section 5.3 we consider the LEXLINRF($\mathbb{Q}$) problem. We observe that applying the algorithm of Section 5.1, which is complete for the integer case, does not result in general in a $LLRF$ for a rational loop, but just what we call a *weak* $LLRF$. This is a $LLRF$ as in Definition 2.11 but changing (11) to $\Delta\rho(\mathrm{x}'') > 0$. It is not immediate that a weak ranking function even implies termination, since $\Delta\rho(\mathrm{x}'')$ can be arbitrarily close to zero. However, we prove that it does, and in fact such a weak ranking function can be converted to a $LLRF$. This provides a complete polynomial-time algorithm for LEXLINRF($\mathbb{Q}$) (which is also optimal with respect to the dimension).

In the rest of this section we assume an input MLC loop specified by the transition polyhedra $\mathcal{Q}_1, \cdots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ is given as a system of inequalities $A\mathrm{x}'' \leq \mathrm{c}''_i$. Since we handle MLC loops, our results apply to SLC loops as a special case; we would like to point out, however, that the coNP-hardness already applies to SLC loops (Section 5.2), and that some interesting examples which demonstrate the advantage of $LLRFs$ over $LRFs$ use just SLC loops (e.g., Example 2.12 on Page 9).

(1) $\mathcal{Q}$ is *totally unimodular* (e.g., DBM). In this case $\mathcal{Q}$ is already integral.

$$\mathit{while}\ (x_1 \le x_2, x_2 \le x_3)\ \mathit{do}\ x_1' \ge x_1 + 1, x_3' \le x_3$$

We compute the LRF $f(x_1, x_2, x_3) = x_3 - x_1$.

(2) $\mathcal{Q}$ is N*-dimensional*. In this case we compute the integer hull of $\mathcal{Q}$.

$$\mathit{while}\ (4x_1 \ge 1)\ \mathit{do}\ 5x_1' \le 2x_1 + 1, 5x_1' \ge 2x_1 - 3$$

Computing the integer hull of $\mathcal{Q}$ adds $-x_1 + x_1' \le -1$ and $\frac{1}{3}x_1 - x_1' \le \frac{1}{3}$. Then we compute the LRF $f(x_1) = x_1 - 1$.

(3) The update is affine linear with integer coefficients, and $\mathcal{C}$ is a *cone*. In this case $\mathcal{Q}$ is already integral.

$$\mathit{while}\ (x_1 + x_2 \ge 0, 2x_2 + x_3 \ge 0)\ \mathit{do}$$
$$x_1' = x_1 - 2x_2 - x_3 - 1, x_2' = x_2, x_3' = x_3$$

We compute the LRF $f(x_1, x_2, x_3) = x_1 + x_2$.

(4) The update is affine linear with integer coefficients, and $\mathcal{C}$ is *totally unimodular*. In this case $\mathcal{Q}$ is already integral.

$$\mathit{while}\ (x_1 \le x_2, x_3 - x_2 \ge 1)\ \mathit{do}\ x_1' = x_1 + x_3 - x_2, x_2' = x_2, x_3' = 2x_3$$

We compute the LRF $f(x_1, x_2, x_3) = x_2 - x_1$.

(5) The update is affine linear with integer coefficients, and $\mathcal{C}$ is N*-dimensional*. In this case we compute the integer hull of $\mathcal{C}$.

$$\mathit{while}\ (-x_1 + x_2 \le 0, -2x_1 - x_2 \ge -1)\ \mathit{do}\ x_1' = x_1, x_2' = x_2 - 2x_1 + 1$$

Computing the integer hull of $\mathcal{C}$ adds $x_1 \ge 1$. Then we compute the LRF $f(x_1, x_2) = 2x_1 + x_2 - 1$.

(6) The update is affine linear with integer coefficients, and $\mathcal{C}$ can be partitioned into independent sets where each is either a *cone*, *totally unimodular*, or N*-dimensional*. In the case of N*-dimensional* we compute its integer hull.

$$\mathit{while}\ (-x_1 + x_2 \le 0, -2x_1 - x_2 \ge -1, x3 \le 1)\ \mathit{do}$$
$$x_1' = x_1, x_2' = x_2 - 2x_1 + x_3, x_3' = x_3$$

$\mathcal{C}$ is partitioned into $T_1 = \{-x_1 + x_2 \le 0,\ \ -2x_1 - x_2 \le -1\}$ and $T_2 = \{x_3 \le 1\}$. T_1 is N*-dimensional* and T_2 is *totally unimodular*. Computing the integer hull of T_1 adds $x_1 \ge 1$. Then we compute the LRF $f(x_1, x_2, x_3) = 2x_1 + x_2 - 1$.

(7) $\mathcal{Q}$ can be partitioned into independent sets that are covered by cases (1)-(6).

$$\mathit{while}\ (4x_1 \ge 1, x_2 \ge 1)\ \mathit{do}\ 2x_1 - 5x_1' \le 3, -2x_1 + 5x_1' \le 1, x_2' = x_2 + 1$$

$\mathcal{C}$ is partitioned into $T_1 = \{4x_1 \ge 1,\ 2x_1 - 5x_1' \le 3,\ -2x_1 + 5x_1' \le 1\}$ and $T_2 = \{x_2 \ge 1, x_2' = x_2 + 1\}$, which are covered by cases (2) and (4). The integer hull of T_1 is as in case (4). Then we compute the LRF $f(x_1, x_2) = x_1 - 1$.

(8) An MLC loop where each path is covered by cases (1)-(7).

Fig. 3. Summary of special PTIME cases of LinRF($\mathbb{Z}$): (1)-(7) summarize the special cases of sections 4.1 and 4.2 for SLC loops; (8) summarizes the special cases of Section 4.5 for MLC loops. Recall that: $\mathcal{Q}$ is the set of constraints that define the loop; $\mathcal{C}$ is the set of constraints that define the loop condition; and N-dimensional means at most N variables, for a fixed N (above we assume $N = 2$).

5.1. A Complete Algorithm for LEXLINRF($\mathbb{Z}$)

The basic building blocks for our *LLRFs* are non-trivial quasi-*LRFs*. These are similar to *LRFs*, except that $\Delta\rho(\mathbf{x}'') > 0$ is not required to hold for all transitions, but rather for at least one.

Definition 5.1. We say that an affine linear function ρ is a quasi-*LRF* for $T \subseteq \mathbb{Q}^{2n}$ if for every $\mathbf{x}'' \in T$ the following holds:

$$\rho(\mathbf{x}) \geq 0 \tag{30}$$

$$\Delta\rho(\mathbf{x}'') \geq 0 \tag{31}$$

We say that it is *non-trivial* if, in addition, inequality (31) is strict, i.e., $\Delta\rho(\mathbf{x}'') > 0$, for at least one $\mathbf{x}'' \in T$.

We say that ρ is a quasi-*LRF* for a rational (respectively integer) loop if it is a quasi-*LRF* for its transition polyhedra (respectively, their integer points).

Example 5.2. Consider the *SLC* loop (12) of Example 2.12: $\rho_1(x_1, x_2, x_3) = x_2$ is a non-trivial quasi-*LRF*; $\rho_2(x_1, x_2, x_3) = x_1$ is not because $\Delta\rho_2(\mathbf{x}'') \geq 0$ does not hold for all transitions; and $\rho_3(x_1, x_2, x_3) = x_3$ is not because $\rho_3(\mathbf{x}) < 0$ for $\mathbf{x} = (2, 1, -1)$. Now consider the *MLC* loop (3) of Section 1: $\rho_4(x_1, x_2) = x_1$ is a non-trivial quasi-*LRF* for both paths of this loop; and $\rho_5(x_1, x_2) = x_2$ is not quasi-*LRF* since $\Delta\rho_5(\mathbf{x}'') \geq 0$ does not hold for all transitions, e.g., it fails for $\mathbf{x}'' = (2, 2, 1, 3)$. Note that ρ_5 is a quasi-*LRF* for the second path, but this is not enough.

Note that when dealing with integer points, we can safely assume that whenever the function decreases in a transition, it decreases at least by 1. In fact, this holds for all affine functions with integer coefficients, and a function with non-integral rational coefficients can always be scaled up to have integer ones.

Our *LLRF* synthesis algorithm is based on repeatedly finding non-trivial quasi-*LRFs*, and therefore we first focus on developing a complete algorithm for synthesizing non-trivial quasi-*LRFs*. The next lemma explains how to represent the space $\mathcal{S}$ of all quasi-*LRFs*, afterwards, we explain how to pick a non-trivial one, if possible, from this space.

LEMMA 5.3. *Given $\mathcal{Q}_1, \dots, \mathcal{Q}_k$, it is possible to build, in polynomial time, a set of inequalities $\mathcal{S}$ whose solutions define the coefficient vectors of all quasi-LRFs for the corresponding transitions $\mathcal{Q}_1 \cup \dots \cup \mathcal{Q}_k$.*

PROOF. Consider the constraints built by the Podelski-Rybalchenko procedure of Theorem 4.20, and change (28e) to $\vec{\eta} \cdot \mathbf{c}'' \leq 0$. Then, these constraints describe the set of all quasi-*LRFs* for $\mathcal{Q}$, rather than *LRFs*. Using the construction of Theorem 4.24, with this change, we get a polyhedron $\mathcal{S}$ of dimension $n' = n+1+\sum_{i=1}^{k} 2m_i$ where m_i is the number of inequalities in $\mathcal{Q}_i$. Assume the first $n+1$ components correspond to the coefficients $(\lambda_0, \vec{\lambda})$ (and the rest correspond to $\vec{\mu}$ and $\vec{\eta}$), then any point $(\lambda_0, \vec{\lambda}, \vec{\mu}, \vec{\eta}) \in \mathcal{S}$ defines a quasi-*LRF* $\rho(\mathbf{x}) = \vec{\lambda} \cdot \mathbf{x} + \lambda_0$ for $\mathcal{Q}_1 \cup \dots \cup \mathcal{Q}_k$. □

The next lemma explains how to pick a non-trivial quasi-*LRF* ρ, if any, from $\mathcal{S}$. Moreover, it shows how to pick one such that $\Delta\rho$ is strict for as many transitions as possible, i.e., there is no other quasi-*LRF* ρ', and valid transition $\mathbf{x}''$, such that $\Delta\rho'(\mathbf{x}'') > 0$ and $\Delta\rho(\mathbf{x}'') = 0$. We refer to such non-trivial quasi-*LRFs* as *optimal*. The importance of this optimal choice is in that it leads to an algorithm that synthesizes *LLRFs* of minimal dimension.

LEMMA 5.4. *There is a polynomial-time algorithm that finds a non-trivial quasi-LRF ρ, if there is any, for $\mathcal{Q}_1,\dots,\mathcal{Q}_k$; moreover, for any quasi-LRF ρ', and valid transition $\mathbf{x}''$, $\Delta\rho(\mathbf{x}'')=0 \Rightarrow \Delta\rho'(\mathbf{x}'')=0$.*

PROOF. The algorithm follows the following steps:

(a) Construct a polyhedron $\mathcal{S}$ of all quasi-$LRFs$ as in Lemma 5.3;
(b) If $\mathcal{S}=\emptyset$ return NONE, otherwise, pick $(\lambda_0,\vec{\lambda},\vec{\mu},\vec{\eta})$ in the *relative interior*[4] of $\mathcal{S}$;
(c) If $\max\{\vec{\lambda}\cdot(\mathbf{x}-\mathbf{x}') \mid \mathbf{x}''\in\mathcal{Q}_i\}>0$, for some $1\le i\le k$, return $\rho(\mathbf{x})=\vec{\lambda}\cdot\mathbf{x}+\lambda_0$, otherwise return NONE.

When the above algorithm returns $\rho\neq$ NONE, it is a non-trivial quasi-LRF since it is a quasi-LRF, and the last step guarantees the existence of at least one $\mathbf{x}''$ for which $\Delta\rho(\mathbf{x}'')>0$. To show completeness of the above algorithm and optimality of ρ, it is enough to show that for any $(\lambda_0',\vec{\lambda}',\vec{\mu}',\vec{\eta}')\in\mathcal{S}$ and $\mathbf{z}''\in\mathcal{Q}_1\cup\cdots\cup\mathcal{Q}_k$, we have $\vec{\lambda}\cdot(\mathbf{z}-\mathbf{z}')=0\Rightarrow\vec{\lambda}'\cdot(\mathbf{z}-\mathbf{z}')=0$.

So, assume that $\vec{\lambda}\cdot(\mathbf{z}-\mathbf{z}')=0$. Define the hyperplane $\mathcal{H}=\{(\alpha_0,\vec{\alpha},\vec{\beta},\vec{\gamma})\in\mathbb{Q}^{n'} \mid \vec{\alpha}\cdot(\mathbf{z}-\mathbf{z}')=0\}$ where $\vec{\alpha}$ is a vector of dimension n, and n' is the dimension of $\mathcal{S}$. By assumption, $(\lambda_0,\vec{\lambda},\vec{\mu},\vec{\eta})\in\mathcal{S}\cap\mathcal{H}$. Note that $\mathcal{S}\cap\mathcal{H}$ is a face of $\mathcal{S}$. If it equals to $\mathcal{S}$, then $(\lambda_0',\vec{\lambda}',\vec{\mu}',\vec{\eta}')\in\mathcal{H}$ and our claim holds. Otherwise, it is a proper face of $\mathcal{S}$. Since $(\lambda_0,\vec{\lambda},\vec{\mu},\vec{\eta})$ was chosen from the relative interior of $\mathcal{S}$, we have $\vec{\lambda}\cdot(\mathbf{z}-\mathbf{z}')>0$, and again our claim holds.

To justify the polynomial-time complexity note that the first step is polynomial by Lemma 5.3; the second step can be done in polynomial time [Schrijver 1986, Cor. 14.1**g**, p. 185]; and the third is also polynomial since it consists of solving at most k LP problems over the rationals. □

Next we observe that finding a non-trivial quasi-LRF for $I(\mathcal{Q}_1)\cup\cdots\cup I(\mathcal{Q}_k)$, i.e., over the integers, can be done by finding one for the corresponding integer hulls.

LEMMA 5.5. *Function ρ a is non-trivial quasi-LRF for $I(\mathcal{Q}_1)\cup\cdots\cup I(\mathcal{Q}_k)$ if and only if it is a non-trivial quasi-LRF for $\mathcal{Q}_{1I}\cup\cdots\cup\mathcal{Q}_{kI}$.*

PROOF. ($\Rightarrow$) Suppose ρ is a non-trivial quasi-LRF for $I(\mathcal{Q}_1)\cup\cdots\cup I(\mathcal{Q}_k)$. Then, since $I(\mathcal{Q}_i)\subseteq\mathcal{Q}_{iI}$, there is an integer point $\mathbf{x}''\in\mathcal{Q}_{1I}\cup\cdots\cup\mathcal{Q}_{kI}$ for which $\Delta\rho(\mathbf{x}'')>0$. It remains to show that for any $\mathbf{x}''\in\mathcal{Q}_{1I}\cup\cdots\cup\mathcal{Q}_{kI}$ we have $\rho(\mathbf{x})\ge 0$ and $\Delta\rho(\mathbf{x}'')\ge 0$. This follows from the fact that, by definition of integer polyhedra, any $\mathbf{x}''\in\mathcal{Q}_{iI}$ is a convex combination of some points from $I(\mathcal{Q}_i)$. ($\Leftarrow$) Suppose ρ is a non-trivial quasi-LRF for $\mathcal{Q}_{1I}\cup\cdots\cup\mathcal{Q}_{kI}$. Then, for any $\mathbf{x}''\in I(\mathcal{Q}_1)\cup\cdots\cup I(\mathcal{Q}_k)$ we have $\rho(\mathbf{x})\ge 0$ and $\Delta\rho(\mathbf{x}'')\ge 0$. It remains to show that there is $\mathbf{x}''\in I(\mathcal{Q}_1)\cup\cdots\cup I(\mathcal{Q}_k)$ for which $\Delta\rho(\mathbf{x}'')>0$. Let $\mathbf{x}''\in\mathcal{Q}_{iI}$ be a point for which $\Delta\rho(\mathbf{x}'')>0$, then, since $\mathbf{x}''$ is a convex combination of some integer points from $I(\mathcal{Q}_i)$, there must be an integer point $\mathbf{z}''\in I(\mathcal{Q}_i)$ for which $\Delta\rho(\mathbf{z}'')>0$. □

Now we are in a position for describing our algorithm for synthesizing a $LLRF$, shown as the procedure `LLRFint` in Algorithm 1. It either returns a $LLRF$ τ or NONE if none exists. Let us first explain the recursive procedure `LLRFSYN`. It builds the $LLRF$ component by component, or more precisely, by finding a suitable first component and calling itself recursively to find the rest. At Line 3 it finds a non-trivial quasi-LRF ρ for the transitions $\mathcal{P}_1\cup\cdots\cup\mathcal{P}_k$. Assuming (as is always safe to do) that the coefficients returned are integer, this ρ ranks all transitions for which $\Delta\rho(\mathbf{x}'')\ge 1$, while for other

[4]For definitions related to faces of polyhedra, and the relative interior, see Section 2.1.

Algorithm 1: Synthesizing Lexicographical Linear Ranking Functions

```
LLRFint($\langle \mathcal{Q}_1, \dots, \mathcal{Q}_k \rangle$)
Input: MLC loop defined by the polyhedra $\mathcal{Q}_1, \dots, \mathcal{Q}_k$
Output: A $LLRF$ for $I(Q_1), \dots, I(Q_k)$, if exists, otherwise NONE
begin
1   Compute the integer hulls $\mathcal{Q}_{1I}, \dots, \mathcal{Q}_{kI}$
2   return LLRFSYN($\langle \mathcal{Q}_{1I}, \dots, \mathcal{Q}_{kI} \rangle$).

LLRFSYN($\langle \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$)
Input: MLC loop defined by the polyhedra $\mathcal{P}_1, \dots, \mathcal{P}_k$
Output: A $LLRF$ for $\mathcal{P}_1, \dots, \mathcal{P}_k$, if exists, otherwise NONE
begin
1   if $\langle \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$ are all empty then return nil

3   else if $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$ has a non-trivial quasi-$LRF$ $\rho$ then
4       $\forall 1 \le i \le k \;.\; \mathcal{P}'_i := \mathcal{P}_i \wedge \Delta\rho(\mathbf{x}'') = 0$
5       $\tau \leftarrow$ LLRFSYN($\langle \mathcal{P}'_1, \dots, \mathcal{P}'_k \rangle$)
6       if $\tau \ne$ NONE then return $\rho{::}\tau$
        else return NONE
8   else return NONE
```

transitions, $\Delta\rho(\mathbf{x}'') = 0$. The set of these transitions is computed at Line 4, and at Line 5 `LLRFSYN` is recursively called in order to find a $LLRF$ τ for them. If it finds one, then it returns $\rho{::}\tau$ as a $LLRF$ for $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$. The recursion stops when all transitions are ranked (Line 2), or when there is no non-trivial quasi-LRF for the current set of transitions (Line 4). An important property of this algorithm is that when calling `LLRFSYN` with integral polyhedra, then the polyhedra passed to the recursive call are also integral. This allows us to rely on Lemmas 5.4 and 5.5, which entail the completeness of the overall algorithm. This also explains why it suffices to compute the integer hulls once, at Line 1 of Procedure `LLRFint`.

Example 5.6. Let us demonstrate the algorithm on the SLC loop (12) of Example 2.12, which is defined by

$$\mathcal{Q} = \{x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge -x_1,\ x'_2 = x_2 - x_1,\ x'_3 = x_3 + x_1 - 2\}.$$

First note that in this case $\mathcal{Q}_I = \mathcal{Q}$ and thus we can skip Line 1 of Procedure `LLRFint`. `LLRFSYN` is first called with $\mathcal{Q}$, and then, at Line 3 it finds the non-trivial quasi-LRF $\rho_1(x_1, x_2, x_3) = x_2$ for $\mathcal{Q}$, at Line 4 it sets $\mathcal{P}'_1$ to $\mathcal{Q} \wedge x_2 - x'_2 = 0$, and at Line 5 `LLRFSYN` is called recursively with this $\mathcal{P}'_1$. Then, at Line 3 it finds the non-trivial quasi-LRF $\rho_2(x_1, x_2, x_3) = x_3$ for $\mathcal{Q} \wedge x_2 - x'_2 = 0$, at Line 4 it sets $\mathcal{P}'_1$ to $\mathcal{Q} \wedge x_2 - x'_2 = 0 \wedge x_3 - x'_3 = 0$ which is an empty polyhedron, and at Line 5 `LLRFSYN` is called recursively with an empty polyhedron. Then, the check at Line 2 succeeds and it returns **nil**. Thus, the final returned value is $\langle x_2, x_3 \rangle$ which is a $LLRF$ for $I(\mathcal{Q}_1)$. Now suppose that we remove $x_3 \ge -x_1$ from $\mathcal{Q}$, and note that we still have $\mathcal{Q}_I = \mathcal{Q}$. Calling `LLRFSYN` with this modified $\mathcal{Q}$ would proceeds as above, however, it would fail to find a non-trivial quasi-LRF for $\mathcal{Q} \wedge x_2 - x'_2 = 0$ and thus it returns NONE. Indeed, in this case $I(\mathcal{Q})$ does not have a $LLRF$ since the loop is non-terminating.

Before formally proving soundness and completeness of Algorithm 1, we state a fundamental observation that we will rely on.

OBSERVATION 5.7. *Let $\mathcal{Q}$ be a transition polyhedron. If ρ is a quasi-LRF for $\mathcal{Q}$, then the points where $\Delta\rho(\mathrm{x}'') = 0$ holds, if any, form a face of $\mathcal{Q}$.*

PROOF. If there is $\mathrm{x}'' \in \mathcal{Q}$ such that $\Delta\rho(\mathrm{x}'') = 0$, then $\min\{\Delta\rho(\mathrm{x}'') \mid \mathrm{x}'' \in \mathcal{Q}\} = 0$. According to the definition of a face, the intersection of the hyperplane $\{\mathrm{x}'' \in \mathbb{Q}^{2n} \mid \Delta\rho(\mathrm{x}'') = 0\}$ with $\mathcal{Q}$ is a face of $\mathcal{Q}$. □

Note that the statement that ρ is non-trivial is equivalent to stating that the face, above, is proper.

LEMMA 5.8. *If* `LLRFint`$(\langle \mathcal{Q}_1, \dots, \mathcal{Q}_k \rangle)$ *returns τ different from* NONE*, then τ is a LLRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$.*

PROOF. We show that when $\mathcal{P}_1, \dots, \mathcal{P}_k$ are integral, and `LLRFSYN`$(\langle \mathcal{P}_1, \dots, \mathcal{P}_k \rangle)$ returns $\tau \neq$ NONE, then τ is a *LLRF* for $I(\mathcal{P}_1), \dots, I(P_k)$. The conclusion of the lemma then follows because `LLRFint` calls `LLRFSYN` with the integer polyhedra $\mathcal{Q}_{1I}, \dots, \mathcal{Q}_{kI}$. The proof is by induction on $\sum \dim(\mathcal{P}_i)$.

Base-case. The base-case is when $\sum \dim(\mathcal{P}_i) = -k$, i.e., all $\mathcal{P}_i$ are empty. In such case the algorithm returns **nil**, which is trivially correct since there are no transitions.

Induction hypothesis. If $\sum \dim(\mathcal{P}_i) < j$, each $\mathcal{P}_i$ is integral, and `LLRFSYN`$(\langle \mathcal{P}_1, \dots, \mathcal{P}_k \rangle)$ returns τ, then τ is a *LLRF* for $I(\mathcal{P}_1), \dots, I(\mathcal{P}_k)$.

Induction step. Assume $\sum \dim(\mathcal{P}_i) = j$, and that `LLRFSYN`$(\langle \mathcal{P}_1, \dots, \mathcal{P}_k \rangle)$ returns $\rho{::}\tau$. Namely, at Line 3 it finds ρ, and $\tau \neq$ NONE is the result of `LLRFSYN`$(\langle \mathcal{P}'_1, \dots, \mathcal{P}'_k \rangle)$ at Line 5. We show that $\rho{::}\tau$ is a *LLRF* for $I(\mathcal{P}_1), \dots, I(\mathcal{P}_k)$. First note the following:

(1) Each $\mathcal{P}'_i$ is integral. This is because it is either empty, or a face of $\mathcal{P}_i$ (by Lemma 5.7), and all faces of an integral polyhedron are integral.
(2) $\sum \dim(\mathcal{P}'_i) < \sum \dim(\mathcal{P}_i) = j$. This is because (*i*) $\forall 1 \leq i \leq k \;.\; \dim(\mathcal{P}'_i) \leq \dim(\mathcal{P}_i)$; and (*ii*) there is $\mathrm{x}'' \in \mathcal{P}_i$, for some i, such that $\Delta\rho(\mathrm{x}'') > 0$, and thus $\mathcal{P}'_i$ is either empty or a proper face of $\mathcal{P}_i$ (by Lemma 5.7), in both cases $\dim(\mathcal{P}'_i) < \dim(\mathcal{P}_i)$.
(3) We may assume that the function ρ has been scaled, if necessary, so that for any $\mathrm{x}'' \in I(\mathcal{P}_1) \cup \cdots \cup I(\mathcal{P}_k)$, either $\Delta\rho(\mathrm{x}'') = 0$ and $\mathrm{x}'' \in I(\mathcal{P}'_1) \cup \cdots \cup I(\mathcal{P}'_k)$, or $\Delta\rho(\mathrm{x}'') \geq 1$.

Using (1,2), we apply the induction hypothesis and conclude that τ is a *LLRF* for $I(\mathcal{P}'_1), \dots, I(\mathcal{P}'_k)$. Using (3) we conclude that $\rho{::}\tau$ is still a *LLRF* for $I(\mathcal{P}'_1), \dots, I(\mathcal{P}'_k)$, and that ρ ranks all transitions of $I(\mathcal{P}_1) \cup \cdots \cup I(\mathcal{P}_k)$ that are not in $I(\mathcal{P}'_1) \cup \cdots \cup I(\mathcal{P}'_k)$. Thus, $\rho{::}\tau$ is a *LLRF* for $I(\mathcal{P}_1), \dots, I(\mathcal{P}_k)$. □

Lemma 5.8 proves that Algorithm 1 is a sound procedure for LEXLINRF($\mathbb{Z}$). In Theorem 5.11 below we combine this with a completeness proof. First, we give sufficient and necessary conditions for the existence of a *LLRF* for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$.

OBSERVATION 5.9. *If there is a LLRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, then every set of transitions $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ has a non-trivial quasi-LRF.*

PROOF. Let $\tau = \langle \rho_1, \dots, \rho_d \rangle$ be a *LLRF* for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$, and T be a set of transitions. Define $I = \{i \mid \mathrm{x}'' \in T \text{ is ranked by } \rho_i\}$, and let $j = \min(I)$. Then, from Definition 2.11, it is easy to verify that ρ_j is a non-trivial quasi-*LRF* for T. □

OBSERVATION 5.10. *If every set of transitions $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ has a non-trivial quasi-LRF, then there is a LLRF for $I(\mathcal{Q}_1), \dots, I(\mathcal{Q}_k)$.*

PROOF. In such case, Algorithm 1, will find a *LLRF*. This is because in every call to `LLRFSYN`, $\mathcal{P}_1, \dots, \mathcal{P}_k$ are integral, and thus, by Lemmas 5.4 and 5.5 the check at Line 3 of `LLRFSYN` is complete. Moreover, the algorithm terminates since $\sum \dim(\mathcal{P}_i)$ decreases in each recursive call and has a lower bound $-k$. □

THEOREM 5.11. *Algorithm 1 is sound and complete for* LEXLINRF($\mathbb{Z}$). *Moreover, when it finds a LLRF, it finds one of a minimal dimension.*

PROOF. If the algorithm returns $\tau = \langle \rho_1, \dots, \rho_d \rangle$, then, by Lemma 5.8, it is a $LLRF$. If it is returns NONE, then it has found a subset of integer points (at Line 3 of `LLRFSYN`) that does not have a non-trivial quasi-LRF. In this case, by Observation 5.9, there is no $LLRF$. Thus, soundness and completeness have been established.

The minimality of the dimension stems from the fact that our algorithm is greedy, i.e., in each step finds (by Lemma 5.4) a $LLRF$ that ranks as many transitions as possible. Assume there is another $LLRF$ $\tau' = \langle \rho'_1, \dots, \rho'_{d'} \rangle$. We show by induction that the set of transitions that are not ranked by $\langle \rho_1, \dots, \rho_i \rangle$, call it $\mathcal{U}_i$, is contained in the set of transitions not ranked by $\langle \rho'_1, \dots, \rho'_i \rangle$, call them $\mathcal{U}'_i$. Observe that since `LLRFSYN` returns immediately if the input polyhedra are empty, we must have $\mathcal{U}_i \neq \emptyset$ for $i \leq d$. It follows that $\mathcal{U}'_i \neq \emptyset$ for $i \leq d$, hence $d' \geq d$.

The claim holds by definition for $i = 0$ since $\mathcal{U}_0 = \mathcal{U}'_0 = I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_1)$. Assume $\mathcal{U}_i \subseteq \mathcal{U}'_i$ for some $0 \leq i < d'$, we show that $\mathcal{U}_{i+1} \subseteq \mathcal{U}'_{i+1}$. Since $\mathcal{U}_i \subseteq \mathcal{U}'_i$ then ρ'_{i+1} is a quasi-LRF for $\mathcal{U}_i$, and since ρ_{i+1} is optimal for $\mathcal{U}_i$, by Lemma 5.4, it cannot be that ρ'_{i+1} ranks a transition from $\mathcal{U}_i$ that is not ranked by ρ_{i+1}, thus $\mathcal{U}_{i+1} \subseteq \mathcal{U}'_{i+1}$. $\square$

The next corollary bounds the dimension of the $LLRF$ inferred by `LLRFSYN` in terms of n, the number of variables in the loop.

COROLLARY 5.12. *If* `LLRFSYN` *returns* $\tau = \langle \rho_1, \dots, \rho_d \rangle$, *then* $d \leq n$.

PROOF. Let $\vec{\lambda}_i$ be the coefficients of ρ_i (i.e., we ignore the constant λ_0); for $1 \leq i \leq d$. We claim that the vectors $\vec{\lambda}_i$ must be linearly independent. Assume the contrary; let i be the first index such that $\vec{\lambda}_i = c_1 \cdot \vec{\lambda}_1 + \cdots + c_{i-1} \cdot \vec{\lambda}_{i-1}$. Pick a transition $\mathbf{x}''$ that is ranked by ρ_i, i.e., $\Delta\rho_i(\mathbf{x}'') = \vec{\lambda}_i \cdot (\mathbf{x} - \mathbf{x}') > 0$ and $\forall 1 \leq j < i \,.\, \Delta\rho_j(\mathbf{x}'') = \vec{\lambda}_j \cdot (\mathbf{x} - \mathbf{x}') = 0$. Then

$$\Delta\rho_i(\mathbf{x}'') = \vec{\lambda}_i \cdot (\mathbf{x} - \mathbf{x}') = (\sum_{j=1}^{i-1} c_j \cdot \vec{\lambda}_j) \cdot (\mathbf{x} - \mathbf{x}') = \sum_{j=1}^{i-1} c_j \cdot \vec{\lambda}_j \cdot (\mathbf{x} - \mathbf{x}') = 0 \qquad (32)$$

which contradicts the assumption that $\Delta\rho_i(\mathbf{x}'') > 0$. Now since each $\vec{\lambda}_i$ is a vector in $\mathbb{Q}^n$, linear independence implies $d \leq n$. $\square$

The above Lemma provides the best bound possible for MLC loops. To see this, consider the MLC loop (3) of Section 1, for which $n = 2$, and note that it has a $LLRF$ with $d = 2$, namely $\langle x_1, x_2 \rangle$, but no $LLRF$ with $d = 1$ (since it does not have a LRF). This can easily be extended to provide an example for any n.

Next, we argue that Procedure `LLRFSYN` can be implemented in polynomial time. Note that this does not mean that LEXLINRF($\mathbb{Z}$) is PTIME-decidable since Algorithm 1 has to compute the integer hulls first, which may take exponential time. However, this does mean that in certain special cases, LEXLINRF($\mathbb{Z}$) is PTIME-decidable.

LEMMA 5.13. *Procedure* `LLRFSYN` *can be implemented in polynomial time.*

PROOF. First note that by Corollary 5.12 the recursion depth is bounded by $n + 1$, and that lines 2 and 3 can be performed in polynomial time in the bit-size of (the current) $\mathcal{P}_1, \dots, \mathcal{P}_n$. However, we cannot immediately conclude that the overall runtime is polynomial since as recursion progresses, the procedure operates on polyhedra obtained by adding additional constraints (at Line 4), that could get bigger and bigger in their bit-size. Thus, to complete the proof, we need to ensure that the bit-size of $\mathcal{P}_1, \dots, \mathcal{P}_n$, at any stage of the recursion, is polynomial in the bit-size of the original ones. Next we show how Line 4 can be implemented to ensure this, exploiting the fact that when $\mathcal{P}_i \wedge \Delta\rho(\mathbf{x}'') = 0$ is not empty, it is a face of $\mathcal{P}_i$.

Recall that any face of $\mathcal{P}_i$ can be obtained by changing some of its inequalities to equalities. Hence, instead of adding $\Delta\rho(\mathbf{x}'') = 0$ to $\mathcal{P}_i$ at Line 4, we can identify those inequalities of $\mathcal{P}_i$ that should be turned into equalities to get $\mathcal{P}_i \wedge \Delta\rho(\mathbf{x}'') = 0$. Changing these inequalities to equalities ensures that the bit-size of $\mathcal{P}_i$, at any stage of the recursion, is at most twice its original bit-size. Finding these inequalities can be done as follows: for each inequality $\mathbf{a} \cdot \mathbf{x} \leq b$ of $\mathcal{P}_i$, we check if $\mathcal{P}_i \wedge \Delta\rho(\mathbf{x}'') = 0 \Rightarrow \mathbf{a} \cdot \mathbf{x} \geq b$ holds, if so, then this inequality should be turned to equality. This check can be done in polynomial time since it is an LP problem and the bit-size of ρ is polynomial in the bit-size of $\mathcal{P}_1, \ldots, \mathcal{P}_n$. □

The above lemma implies that, as for LINRF($\mathbb{Z}$), if it is guaranteed that the transition polyhedra are integral, or their integer hull can be computed in polynomial time, then the LEXLINRF($\mathbb{Z}$) problem can be solved in polynomial time.

THEOREM 5.14. *The* LEXLINRF($\mathbb{Z}$) *problem for MLC loops is PTIME-decidable if each path corresponds to one of the special cases discussed in sections 4.1 and 4.2.*

PROOF. For those special cases either we do not compute the integer hulls since they are already integral, or we compute them in polynomial time. Then Algorithm 1 becomes polynomial-time since Line 1 of `LLRFint` can be done in polynomial time, and `LLRFSYN` is polynomial according to Lemma 5.13. □

It may be worthwhile to point out that even if we do not have a PTIME-decidable case, we can always apply Procedure `LLRFSYN` to the given polyhedra—if it produces a $LLRF$, we have a sound result in polynomial time.

5.2. Complexity of LEXLINRF($\mathbb{Z}$)

In this section we show that the LEXLINRF($\mathbb{Z}$) problem, in the general case, is coNP-complete. First, coNP-hardness follows from the coNP-hardness of LINRF($\mathbb{Z}$) as in Theorem 3.1. This is because the construction in Theorem 3.1 either produces a loop that has a LRF (which is also a $LLRF$) or else it is non-terminating (so it does not have any kind of ranking function). For the inclusion in coNP, we show that the complement problem, i.e., the nonexistence of a $LLRF$, has a polynomially checkable witness.

COROLLARY 5.15. *There is no LLRF for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, if and only if there is $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ for which there is no non-trivial quasi-LRF.*

PROOF. Immediate from Observations 5.9 and 5.10. □

The above observation suggests that such T can be used as a witness, however, T might include infinite number of transitions, and thus it does not immediately meet our needs (polynomially checkable witness).

Example 5.16. We show a case in which T must consist of infinitely many points. Let $\mathcal{Q} = \{x' \leq x - 1\}$ and take an arbitrary finite $T \subseteq \mathcal{Q}$. Now define $\lambda_0 = \min\{x \mid (x, x') \in T\}$, then $\rho(x) = x - \lambda_0$ is a non-trivial quasi-LRF (actually LRF) for T and thus T does not prove that there is no quasi-LRF for $\mathcal{Q}$. Any set of transitions out of $\mathcal{Q}$ that does not have a quasi-LRF must be infinite.

To overcome this finiteness problem, we use notions similar to the witness and h-witness that we have used for the case of LINRF($\mathbb{Z}$). In particular, we show that the existence of T as in Corollary 5.15 can be witnessed by finite sets $X \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ and $Y \subseteq I(\mathcal{R}_{\mathcal{Q}_1}) \cup \cdots \cup I(\mathcal{R}_{\mathcal{Q}_k})$, whose bit-size is bounded polynomially in the bit-size of the input.

Definition 5.17. Let $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$, such that (*i*) $X_i \subseteq I(\mathcal{Q}_i)$; (*ii*) $Y_i \subseteq I(\mathcal{R}_{\mathcal{Q}_i})$; and (*iii*) $Y_i \neq \emptyset \Rightarrow X_i \neq \emptyset$. We say that X and Y form a witness against the existence of a *LLRF* for $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$, if the following set of linear constraints, denoted by $\Phi(X,Y)$, has no solution

$$\vec{\lambda}\cdot\mathbf{x} + \lambda_0 \geq 0 \quad \text{for all } \mathbf{x}'' \in X \tag{33a}$$

$$\vec{\lambda}\cdot\mathbf{y} \geq 0 \quad \text{for all } \mathbf{y}'' \in Y \tag{33b}$$

$$\vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') \geq 0 \quad \text{for all } \mathbf{x}'' \in X \tag{33c}$$

$$\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') \geq 0 \quad \text{for all } \mathbf{y}'' \in Y \tag{33d}$$

$$\sum_{\mathbf{x}'' \in X} \vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') + \sum_{\mathbf{y}'' \in Y} \vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') \geq 1 \tag{33e}$$

LEMMA 5.18. *Let $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$ be as in Definition 5.17. Then there is $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ that has no non-trivial quasi-LRF.*

PROOF. We construct such T. First note that for $\mathbf{x}'' \in X_i$ and $\mathbf{y}'' \in Y_i$, the point $\mathbf{x}'' + a\mathbf{y}''$, for any integer $a \geq 0$, is a transition in $I(\mathcal{Q}_i)$. Now define

$$T = \{\mathbf{x}'' + a\mathbf{y}'' \mid \mathbf{x}'' \in X_i, \mathbf{y}'' \in Y_i,\ \text{integer } a \geq 0\ \}\ .$$

Clearly $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$. We claim that T has no non-trivial quasi-*LRF*. Assume the contrary, i.e., there is $(\lambda_0,\vec{\lambda}) \in \mathbb{Q}^{n+1}$ such that $\rho(\mathbf{x}) = \vec{\lambda}\cdot\mathbf{x} + \lambda_0$ is a non-trivial quasi-*LRF* for T. We show that $(c\lambda_0,c\vec{\lambda})$, for *some* $c > 0$, is a solution of $\Phi(X,Y)$, which contradicts the assumption that X and Y form a witness as in Definition 5.17.

We first show that (33a–33d) of $\Phi(X,Y)$ hold for $(c\lambda_0,c\vec{\lambda})$ with *any* $c > 0$. Pick arbitrary $\mathbf{x}'' \in X_i$ and $\mathbf{y}'' \in Y_i$. Since ρ is a non-trivial quasi-*LRF* for T, inequalities (30,31) on Page 28 must hold for $\mathbf{x}'' + a\mathbf{y}'' = \left(\begin{smallmatrix}\mathbf{x}+a\mathbf{y}\\ \mathbf{x}'+a\mathbf{y}'\end{smallmatrix}\right) \in T$. Namely, the following must hold for any integer $a \geq 0$

$$\rho(\mathbf{x} + a\mathbf{y}) = \vec{\lambda}\cdot(\mathbf{x} + a\mathbf{y}) + \lambda_0 = \vec{\lambda}\cdot\mathbf{x} + \lambda_0 + a\vec{\lambda}\cdot\mathbf{y} \geq 0 \tag{34}$$

$$\Delta\rho(\mathbf{x}'' + a\mathbf{y}'') = \vec{\lambda}\cdot(\mathbf{x} + a\mathbf{y}) - \vec{\lambda}\cdot(\mathbf{x}' + a\mathbf{y}') = \vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') + a\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') \geq 0 \tag{35}$$

This implies

(*i*) $\vec{\lambda}\cdot\mathbf{x} + \lambda_0 \geq 0$, otherwise (34) is false for $a = 0$;
(*ii*) $\vec{\lambda}\cdot\mathbf{y} \geq 0$, otherwise (34) is false for $a > -(\vec{\lambda}\cdot\mathbf{x} + \lambda_0)/(\vec{\lambda}\cdot\mathbf{y})$;
(*iii*) $\vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') \geq 0$, otherwise (35) is false for $a = 0$; and
(*iv*) $\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') \geq 0$, otherwise (35) is false for $a > -\vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}')/\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}')$.

Note that the inequalities in (i–iv) above are those used in (33a–33d). Hence (33a–33d) hold for $(\lambda_0,\vec{\lambda})$, and clearly, also for $(c\lambda_0,c\vec{\lambda})$ with any $c > 0$.

Now we show that (33e) of $\Phi(X,Y)$ holds for $(c\lambda_0,c\vec{\lambda})$, for *some* $c > 0$. Since ρ is a non-trivial quasi-*LRF*, then, inequality (31) must be strict for at least one $\mathbf{x}'' + a\mathbf{y}'' = \left(\begin{smallmatrix}\mathbf{x}\ +a\mathbf{y}\\ \mathbf{x}'+a\mathbf{y}'\end{smallmatrix}\right) \in T$, i.e., $\Delta\rho(\mathbf{x}'' + a\mathbf{y}'') = \vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') + a\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') > 0$. This means that either $\vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') > 0$ or $\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') > 0$ must hold. Taking $c > 0$ large enough, we have $c\vec{\lambda}\cdot(\mathbf{x} - \mathbf{x}') \geq 1$ or $c\vec{\lambda}\cdot(\mathbf{y} - \mathbf{y}') \geq 1$. Thus, inequality (33e) holds for $(c\lambda_0,c\vec{\lambda})$. Since (33a–33d) also hold for this $(c\lambda_0,c\vec{\lambda})$, it is a solution of $\Phi(X,Y)$. $\square$

LEMMA 5.19. *If there is $T \subseteq I(\mathcal{Q}_1) \cup \cdots \cup I(\mathcal{Q}_k)$ that has no non-trivial quasi-LRF, then there are finite sets $X = X_1 \cup \cdots \cup X_k$ and $Y = Y_1 \cup \cdots \cup Y_k$, fulfilling the conditions of Definition 5.17.*

PROOF. Let $\mathbf{x}''$ be an arbitrary member of T. Let $\mathcal{Q} \in \{\mathcal{Q}_1, \dots, \mathcal{Q}_k\}$ so that $\mathbf{x}'' \in \mathcal{Q}$, and consider the generator representation

$$\mathcal{Q}_I = \text{convhull}\{\mathbf{x}''_1, \dots, \mathbf{x}''_m\} + \text{cone}\{\mathbf{y}''_1, \dots, \mathbf{y}''_t\}\,.$$

Using these representation, we have $\mathbf{x}'' = \sum_{i=1}^{m} a_i \mathbf{x}''_i + \sum_{j=1}^{t} b_j \mathbf{y}''_j$ for some rationals $a_i, b_j \geq 0$, and $\sum_i a_i = 1$. We let $ver(\mathbf{x}'')$ be the set of all vertices $\mathbf{x}''_i$ with $a_i > 0$ and $rays(\mathbf{x}'')$ be the set of all rays $\mathbf{y}''_j$ with $b_j > 0$.

For $\ell = 1, \dots, k$, define $X_\ell = \cup\{ver(\mathbf{x}'') \mid \mathbf{x}'' \in T \cap I(\mathcal{Q}_\ell)\}$ and $Y_\ell = \cup\{rays(\mathbf{x}'') \mid \mathbf{x}'' \in T \cap I(\mathcal{Q}_\ell)\}$. Next we show that $X = X_1 \cup \dots \cup X_k$ and $Y = Y_1 \cup \dots \cup Y_k$ form a witness as in Definition 5.17.

Conditions (i,ii) of Definition 5.17 hold by construction, and Condition (iii) holds because $\sum_i a_i = 1$. What is left to show is that $\Phi(X, Y)$ has no solution. Assume the contrary, i.e., $\Phi(X, Y)$ has a solution $(\lambda_0, \vec{\lambda}) \in \mathbb{Q}^{n+1}$. We claim that then $\rho(\mathbf{x}) = \vec{\lambda}\cdot\mathbf{x} + \lambda_0$ is a non-trivial quasi-LRF for T, which contradicts the lemma's assumption. Pick an arbitrary $\mathbf{x}'' \in T$ and write it, using the corresponding X_ℓ and Y_ℓ, as $\mathbf{x}'' = \sum_{i=1}^{m} a_i \mathbf{x}''_i + \sum_{j=1}^{t} b_j \mathbf{y}''_j$ where $a_i, b_j > 0$ and $\sum_{i=1}^{m} a_i = 1$. Since (33a,33c) hold for each $\mathbf{x}''_i \in X$ and (33b,33d) hold for each $\mathbf{y}''_j \in Y$, we have

$$\begin{aligned}
\rho(\mathbf{x}) =& \vec{\lambda} \cdot (\sum_{i=1}^{m} a_i \mathbf{x}_i + \sum_{j=1}^{t} b_j \mathbf{y}_j) + \lambda_0 \\
=& \sum_{i=1}^{m} a_i \cdot (\vec{\lambda}\cdot\mathbf{x}_i + \lambda_0) + \sum_{j=1}^{t} b_j \vec{\lambda}\cdot\mathbf{y}_j \geq 0
\end{aligned}$$

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

Thus, ρ satisfies (30,31) for any $\mathbf{x}'' \in T$. Now since (33e) holds, there must be $\mathbf{x}''_i \in X$ or $\mathbf{y}''_j \in Y$ for which $\vec{\lambda} \cdot (\mathbf{x}_i - \mathbf{x}'_i) > 0$ or $\vec{\lambda} \cdot (\mathbf{y}_j - \mathbf{y}'_j) > 0$. Now note that since X and Y were constructed from the vertices and rays of the transitions in T, these $\mathbf{x}''_i$ or $\mathbf{y}''_j$ must correspond to some $\mathbf{x}'' \in T$, and thus it must be the case that $\Delta\rho(\mathbf{x}'') > 0$ for this specific $\mathbf{x}''$, i.e., inequality (31) is strict for $\mathbf{x}''$. $\square$

Example 5.20. For $\mathcal{Q} = \{x' \leq x - 1\}$ of Example 5.16, we claim that $X = \{(0, -1)\}$ and $Y = \{(1, 1), (-1, -1)\}$ form a witness as in Definition 5.17. It is easy to check that X and Y satisfy conditions (i–iii). Then, $\Phi(X, Y)$ is the set of inequalities $\{\lambda_0 \geq 0,\ \lambda_1 \geq 0,\ -\lambda_1 \geq 0,\ \lambda_1 \geq 1\}$ which has no solution.

Example 5.21. Consider an MLC loop represented by

$$\begin{aligned}
\mathcal{Q}_1 &= \{x_1 \geq 0,\ x_2 \geq 0,\ x'_1 = x_1 - 1\} \\
\mathcal{Q}_2 &= \{x_1 \geq 0,\ x_2 \geq 0,\ x'_2 = x_2 - 1\}
\end{aligned}$$

and let

$$\begin{aligned}
X_1 &= \{(0, 0, -1,\ 0)\},\ & Y_1 &= \{(0, 0, 0, 1)\}, \\
X_2 &= \{(0, 0,\ 0, -1)\},\ & Y_2 &= \{(0, 0, 1, 0)\}.
\end{aligned}$$

We claim that these sets form a witness as in Definition 5.17. It is easy to check that they satisfy conditions (i–iii) of Definition 5.17. Substituting these points in (33e) gives $0 \geq 1$, so clearly (33a–33e) are unsatisfiable.

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

LEMMA 5.22. *If there is a finite witness for the nonexistence of LLRF for* $I(\mathcal{Q}_1), \ldots, I(\mathcal{Q}_k)$*, then there is one defined by* $X = X_1 \cup \cdots \cup X_k$ *and* $Y = Y_1 \cup \cdots \cup Y_k$ *such that* $\sum_{i=1}^{k} |X_i| + |Y_i| \leq 6n+2$*; and its bit-size is polynomial in the bit-size of* $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$.

PROOF. Consider the witness constructed in Lemma 5.19, and recall that $\Phi_1 = \Phi(X, Y)$ has no solution. Let Z be any maximal linearly-independent subset of $X \cup Y$. Clearly, $|Z| \leq 2n$. Let Φ_2 be the formula obtained from Φ_1 by replacing (33e) with

$$\sum_{\mathbf{z}'' \in Z} \vec{\lambda} \cdot (\mathbf{z} - \mathbf{z}') \geq 1 \tag{36}$$

We claim that Φ_2 has no solution. To see this, take arbitrary $(\lambda_0, \vec{\lambda}) \in \mathbb{Q}^{2n}$, we know it is not a solution of Φ_1. If this is because one of the inequalities in (33a-33d) is false, then it is clearly not a solution of Φ_2 since it includes all such inequalities. If all inequalities in (33a-33d) are true, then (33e) must be false. Since all terms in the sum are non-negative, they must all be zero, that is, $\vec{\lambda} \cdot (\mathbf{z} - \mathbf{z}') = 0$ for any $\mathbf{z}'' \in X \cup Y$. Otherwise, $(c\lambda_0, c\vec{\lambda})$ for $c \geq 1$ large enough would be a solution of Φ_1. Thus, inequality (36) is false.

A corollary of Farkas' Lemma [Schrijver 1986, p. 94] states that: if a set of inequalities over $\mathbb{Q}^d$ has no solution, there is a subset of at most $d+1$ inequalities that has no solution. Let Φ_3 be such a subset of Φ_2, it has at most $n+2$ inequalities (since Φ_2 is over $\mathbb{Q}^{n+1}$). Note that Φ_3 must include inequality (36), otherwise it is trivially satisfiable. Let $X' = X'_1 \cup \ldots \cup X'_k \subseteq X$ and $Y' = Y'_1 \cup \ldots \cup Y'_k \subseteq Y$ be the points involved in the inequalities of Φ_3 (including (36)), then $\sum_{i=1}^{k} |X'_i| + |Y'_i| \leq n+1+2n = 3n+1$. To get a witness as per Definition 5.17, if, for any $i \leq k$, $Y'_i \neq \emptyset$ and $X'_i = \emptyset$, we include an arbitrary point $\mathbf{x}'' \in X_i$ to X'_i. This can at most double the size of these sets, i.e., $\sum_{i=1}^{k} |X'_i| + |Y'_i| \leq 6n+2$ (or $\sum_{i=1}^{k} |X'_i| + |Y'_i| \leq 3n+1+k$ when $k < 3n+1$).

We claim that $\langle X', Y' \rangle$ is a witness that fulfills the conditions of Definition 5.17. It satisfies conditions (i-iii) by construction. Next, we show that $\Phi_4 = \Phi(X', Y')$ has no solution. Take arbitrary $(\lambda_0, \vec{\lambda}) \in \mathbb{Q}^{n+1}$, we know it is not a solution for Φ_2. If it is because one of the inequalities in (33a-33d) is false, then it is clearly not a solution of Φ_4 since it includes all such inequalities. If all inequalities in (33a-33d) are true, then (36) must be false, and then we must have $\vec{\lambda} \cdot (\mathbf{z} - \mathbf{z}') = 0$ for any $\mathbf{z}'' \in Z$. Now since any $\mathbf{z}'' \in X' \cup Y'$ is a linear combination of points from Z, $\vec{\lambda} \cdot (\mathbf{x} - \mathbf{x}') = 0$ for any $\mathbf{x}'' \in X'$ and $\vec{\lambda} \cdot (\mathbf{y} - \mathbf{y}') = 0$ for any $\mathbf{y}'' \in Y'$. Thus, inequality (33e) of Φ_4 is false.

Finally, we show that the bit-size of the witness is polynomial in the bit-size of the input. Recall that the points of X' and Y' come from the generator representations of $\mathcal{Q}_{1I}, \ldots, \mathcal{Q}_{kI}$, and that there is a generator representation for each $\mathcal{Q}_{iI}$ in which each vertex/ray can fit in $\|\mathcal{Q}_{iI}\|_v$ bits. Thus, the bit-size of X' and Y' is bounded by $(6n+2) \cdot \max_i \|\mathcal{Q}_{iI}\|_v$. By Theorem 2.8, since the dimension of each $\mathcal{Q}_i$ is $2n$,

$$(6n+2) \cdot \max_i \|\mathcal{Q}_{iI}\|_v \leq (6n+2) \cdot (6 \cdot (2n)^3 \cdot \max_i \|\mathcal{Q}_i\|_f) \leq (288n^4 + 96n^3) \cdot \max_i \|\mathcal{Q}_i\|_b$$

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

THEOREM 5.23. LEXLINRF($\mathbb{Z}$) $\in$ coNP *for MLC loops.*

PROOF. We show that the complement of LINRF($\mathbb{Z}$) has a polynomially checkable witness. The witness is a listing of sets of integer points $X = X_1 \cup \cdots \cup X_k$ and $Y =$

$Y_1 \cup \cdots \cup Y_k$ of at most $6n+2$ elements and has a polynomial bit-size (specifically, a bit-size bounded as in Lemma 5.22). Verifying a witness consists of the following steps:

Step 1. Verify that each $\mathbf{x}'' \in X_i$ is in $I(\mathcal{Q}_i)$, which can be done by verifying $A_i''\mathbf{x}'' \leq \mathbf{c}_i''$; and that each $\mathbf{y}'' \in Y_i$ is in $I(\mathcal{R}_Q)$, which can be done by verifying $A_i''\mathbf{y}'' \leq 0$. This is done in polynomial time. Note that according to Lemma 5.18 it is not necessary to check that X and Y come from a particular generator representation.

Step 2. Verify that $\Phi(X, Y)$ has no solutions, which can be done in polynomial time since it is an LP problem over $\mathbb{Q}^{n+1}$. $\square$

5.3. Lexicographic Ranking Functions over the Rationals

In this section we address the LEXLINRF($\mathbb{Q}$) problem. In particular, we show that Procedure LLRFSYN, when applied to the input polyhedra $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ instead of their integer hulls, can be used to decide the existence of a $LLRF$ for $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$. However, in such case, the returned value $\tau = \langle \rho_1, \ldots, \rho_d \rangle$ of the algorithm does not fit in the class of $LLRFs$ as in Definition 2.11. We define a new class of $LLRFs$ that captures such functions, and prove that it is actually equivalent to that of Definition 2.11 as far as the existence of a $LLRF$ is concerned.

First recall that in Section 2.4 we discussed the possibility of replacing inequality $\Delta\rho_i(\mathbf{x}'') \geq 1$ by $\Delta\rho_i(\mathbf{x}'') \geq \delta_i$ in condition (11) of Definition 2.11. With this change, $\tau = \langle \rho_1, \ldots, \rho_d \rangle$ is a $LLRF$ if and only if there are positive $\delta_1, \ldots, \delta_d$ such that, for any $\mathbf{x}'' \in \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_k$ there exists i for which the following hold

$$\forall j < i \ . \ \Delta\rho_j(\mathbf{x}'') \geq 0 \tag{37}$$

$$\forall j \leq i \ . \quad \rho_j(\mathbf{x}) \geq 0 \tag{38}$$

$$\Delta\rho_i(\mathbf{x}'') \geq \delta_i \tag{39}$$

This is equivalent to Definition 2.11, as far as the existence of a $LLRF$ is concerned, since $c\tau$, for any $c > \min(\delta_i)^{-1}$, is a corresponding $LLRF$ as in Definition 2.11. In the rest of this section, for the sake of simplifying the formal presentation, we use this notion of $LLRFs$.

Let us start by explaining why the returned value of Procedure LLRFSYN, in the rational case, does not fit in the above class of $LLRFs$. For this, let us consider a non-trivial quasi-LRF ρ synthesized at Line 3. In the integer case, all integer transitions of $\mathcal{P}_1, \ldots, \mathcal{P}_k$ that do not pass to $\mathcal{P}_1', \ldots, \mathcal{P}_k'$ are ranked by this ρ. This is because $\Delta\rho(\mathbf{x}'') \geq 1$ for all such transitions (see the proof of Lemma 5.8, point (3)). This, however, is not true when considering rational transitions. In this case, all transitions that do not pass to $\mathcal{P}_1', \ldots, \mathcal{P}_k'$ satisfy $\Delta\rho(\mathbf{x}'') > 0$, but it is not guaranteed that $\Delta\rho(\mathbf{x}'')$ has a minimum δ over this set of transitions. For example, take $\mathcal{P}_1 = \{x \geq 0, x = 2x'\}$ and $\rho(x) = x$, then $\mathcal{P}_1' = \{x = 0, x' = 0\}$. The transitions that do not pass to $\mathcal{P}_1'$ are those specified by the non-closed polyhedron $\{x > 0, x = 2x'\}$, in which $\Delta\rho$ does not have a positive lower bound. This leads us to introduce *weak LLRFs*.

Definition 5.24. We say that $\tau = \langle \rho_1, \ldots, \rho_d \rangle$ is a *weak LLRF* for $\mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_k$, if and only if for any $\mathbf{x}'' \in \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_k$ there exists i for which (37,38) hold, as well as

$$\Delta\rho_i(\mathbf{x}'') > 0 \tag{40}$$

(which replaces (39)).

While any $LLRF$ is a also weak $LLRF$, the converse is more subtle. Over the integers, the existence of a weak $LLRF$ implies the existence of a $LLRF$ (since $\Delta\rho_i(\mathbf{x}'') > 0$ means $\Delta\rho_i(\mathbf{x}'') \geq 1$ when the coefficients and state variables are integer). Over the rationals, such an implication is not immediate. Moreover, even whether a weak ranking function implies termination is unclear, as infinitely descending sequences of positive rationals exist.

Example 5.25. Consider the following MLC loop

$$\begin{array}{lllll} loop: & \{x_1 \geq 0, & x_1' = x_1 - 1\} & & \\ \vee & \{x_1 \geq 0,\ x_2 - x_1 \geq 0, & x_1' = x_1, & x_2' = x_2 - 1\} & \\ \vee & \{x_1 \geq 0,\ x_2 - x_1 \geq 0,\ x_3 \geq 0, & x_1' \leq \frac{1}{2}x_1, & x_2' = x_2, & x_3' = x_3 - 1\} \end{array} \quad (41)$$

Applying Procedure LLRFSYN to the corresponding transition polyhedra $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3$ possibly returns $\tau = \langle x_1, x_2 - x_1, x_3 \rangle$. It is easy to see that it is a weak $LLRF$ over the rationals, and, consequently, it is a $LLRF$ over the integers. To see why it is not a $LLRF$ over the rationals, assume the first component of τ decreases by at least $\delta_1 > 0$. All transitions for which $x_1 - x_1' < \delta_1$ are not ranked by this component and thus should be ranked by either the second or the third. Let us take $\mathbf{x}'' \in \mathcal{Q}_3$ such that $\mathbf{x} = (\delta_1, 1, 1)$ and $\mathbf{x}' = (\frac{1}{2}\delta_1, 1, 0)$. This transition is not ranked by the first component since $\Delta\rho_1(\mathbf{x}'') = \frac{1}{2}\delta_1 < \delta_1$, and it is not ranked by the second or the third since $\Delta\rho_2(\mathbf{x}'') = -\frac{1}{2}\delta_1 < 0$. Nonetheless, this loop is terminating over the rationals and has a $LLRF$, and later we show how to obtain it.

Over the rationals, Procedure LLRFSYN is sound and complete for synthesizing weak $LLRFs$. Moreover, as in the integer case, it synthesizes one with minimal dimension.

LEMMA 5.26. *Procedure* LLRFSYN*, when applied to* $\mathcal{Q}_1, \cdots, \mathcal{Q}_k$*, is sound and complete for the existence of a weak* $LLRF$ *for* $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$*. Moreover, if* LLRFSYN$(\langle \mathcal{Q}_1, \ldots, \mathcal{Q}_k \rangle)$ *returns* τ *different from* NONE*, then* τ *is a weak* $LLRF$ *of minimal dimension for* $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$.

PROOF. Suppose that LLRFSYN$(\langle \mathcal{Q}_1, \ldots, \mathcal{Q}_k \rangle)$ returns τ. Then, as in the proof of Lemma 5.8, we can show that τ is a weak $LLRF$. We prefer not to repeat the whole proof but just indicate the difference, which boils down to drop points (1) and (3) regarding the integrality of corresponding polyhedra and a non-zero decrease being at least 1.

This gives soundness; for completeness, the proof is as that of Theorem 5.11. In fact, the sufficient and necessary condition for the existence of a $LLRF$, stated in Observations 5.9 and 5.10, is a condition for existence of a weak $LLRF$ when applied to $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$.

The minimality follows from the same consideration as in the proof of Theorem 5.11. □

In the rest of this section we show how one can construct a $LLRF$ for $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ from a weak $LLRF$. This implies soundness and completeness of Procedure LLRFSYN as a decision procedure for LEXLINRF($\mathbb{Q}$), and its usage for synthesis of $LLRFs$. To simplify notation, we shall consider the polyhedra $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ to be fixed up to the completion of the proof.

Here is a brief outline of the construction. The first step, culminating in Lemma 5.29, shows how to transform the $LLRF$ $\langle \rho_1, \ldots, \rho_d \rangle$ into another one $\langle f_1, \ldots, f_d \rangle$, where each f_i will be a linear combination of ρ's, so that if component i is used for ranking some transition of one of the transition polyhedron $\mathcal{Q}_\ell$, we will be ensured that f_i is non-decreasing over all of this $\mathcal{Q}_\ell$ (even over transitions that are already ranked by a previous component). Consequently, in Lemmas 5.32 and 5.33, we show how thanks to this property, the ranking-function components can be "nudged" so that the weak $LLRF$ becomes a proper one.

Definition 5.27. Let $\tau = \langle \rho_1, \dots, \rho_d \rangle$ be a weak *LLRF* for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$. The *ranking chain* for τ is the $(d+1)$-tuple of sets, $U_1, \dots, U_{d+1}$, defined by $U_1 = \mathcal{Q}_1 \cup \dots \cup \mathcal{Q}_k$, and $U_{i+1} = U_i \wedge (\Delta\rho_i(\mathbf{x}'') = 0)$.

Observe that

$$\mathcal{Q}_1 \cup \dots \cup \mathcal{Q}_k = U_1 \supseteq U_2 \supseteq \dots \supseteq U_d \supseteq U_{d+1} = \emptyset.$$

It is easy to see that if for some j, $U_j = U_{j+1}$, it is possible to omit ρ_j from τ without any harm. We say that τ is *irredundant* if

$$\mathcal{Q}_1 \cup \dots \cup \mathcal{Q}_k = U_1 \supset U_2 \supset \dots \supset U_d \supset U_{d+1} = \emptyset. \tag{42}$$

OBSERVATION 5.28. *A weak LLRF computed by Procedure* LLRFSYN *is irredundant. In fact, U_i is the union $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$ of the arguments to the i-th recursive call.*

By the definition of a weak *LLRF*, and the definition of $U_1, \dots, U_{d+1}$, the following properties clearly follow:

$$\forall \mathbf{x}'' \in U_i \ . \quad \rho_i(\mathbf{x}) \geq 0, \tag{43_i}$$
$$\forall \mathbf{x}'' \in U_i \setminus U_{i+1} \ . \ \Delta\rho_i(\mathbf{x}'') > 0, \tag{44_i}$$
$$\forall \mathbf{x}'' \in U_{i+1} \ . \ \Delta\rho_i(\mathbf{x}'') = 0\,. \tag{45_i}$$

Note that each U_i is a finite union of closed polyhedra, obtained by intersecting U_1 with some hyperplanes. For $1 \leq i \leq d$, let $J_i = \{j \mid \mathcal{Q}_j \cap U_i \neq \emptyset\}$, and let $\overline{U}_i = \bigcup_{j \in J_i} \mathcal{Q}_j$. This means that if U_i includes a point from $\mathcal{Q}_j$, then $\overline{U}_i$ includes all points of $\mathcal{Q}_j$. Note that $\overline{U}_i \supseteq \overline{U}_{i+1}$. The next lemma shows that one can construct, for each U_i, a function f_i such that the domain on which (44_i) holds is extended to $\overline{U}_i \setminus U_{i+1}$. These functions are later used in constructing a *LLRF* for $\mathcal{Q}_1, \dots, \mathcal{Q}_k$.

LEMMA 5.29. *Given an irredundant weak LLRF, τ, and its ranking chain $\{U_i\}$, one can construct, for each $1 \leq i \leq d$, an affine function $f_i : \mathbb{Q}^n \to \mathbb{Q}$ such that*

$$\forall \mathbf{x}'' \in U_i \ . \ f_i(\mathbf{x}) \geq \rho_i(\mathbf{x}) \geq 0 \tag{46_i}$$
$$\forall \mathbf{x}'' \in \overline{U}_i \setminus U_{i+1} \ . \quad \Delta f_i(\mathbf{x}'') > 0 \tag{47_i}$$
$$\forall \mathbf{x}'' \in U_{i+1} \ . \quad \Delta f_i(\mathbf{x}'') = 0\,. \tag{48_i}$$

PROOF. The proof proceeds by induction.

Base-case. For the base-case we take $i = 1$, and define $f_1(\mathbf{x}) = \rho_1(\mathbf{x})$. Since $\overline{U}_1 = U_1$, (46_1–48_1) hold (they are equivalent to (43_1–45_1) in this case).

Induction hypothesis. Let $1 \leq i < d$, and assume that $f_1, \dots, f_i$ have been defined. In particular, f_i satisfies (46_i–48_i). Only f_i is used in the induction step below.

Induction step. We show that $f_{i+1}(\mathbf{x}) = \rho_{i+1}(\mathbf{x}) + (\xi + 1) \cdot f_i(\mathbf{x})$, for some $\xi \geq 0$, satisfies (46_{i+1}–48_{i+1}). Most of the proof deals with finding ξ and constructing some related properties. Consider $\mathbf{x}'' \in \overline{U}_{i+1}$. If $\mathbf{x}'' \in U_{i+1}$ then by (48_i) we have $\Delta f_i(\mathbf{x}'') = 0$, and if $\mathbf{x}'' \notin U_{i+1}$ then $\mathbf{x}'' \in \overline{U}_{i+1} \setminus U_{i+1} \subseteq \overline{U}_i \setminus U_{i+1}$ and by (47_i) we have $\Delta f_i(\mathbf{x}'') > 0$. This means that the conjunction $\mathbf{x}'' \in \overline{U}_{i+1} \wedge \Delta f_i(\mathbf{x}'') \leq 0$ refers only to the points of U_{i+1}, and such points, by (44_{i+1},45_{i+1}), satisfy $\Delta\rho_{i+1}(\mathbf{x}'') \geq 0$. Thus, we get

$$\mathbf{x}'' \in \overline{U}_{i+1} \wedge \Delta f_i(\mathbf{x}'') \leq 0 \Rightarrow \Delta\rho_{i+1}(\mathbf{x}'') \geq 0\,. \tag{49}$$

Take $j \in J_{i+1}$, since $\mathcal{Q}_j \subseteq \overline{U}_{i+1}$, (49) still holds when replacing $\overline{U}_{i+1}$ by $\mathcal{Q}_j$

$$\mathbf{x}'' \in \mathcal{Q}_j \wedge \Delta f_i(\mathbf{x}'') \le 0 \Rightarrow \Delta\rho_{i+1}(\mathbf{x}'') \ge 0\,. \tag{50}$$

Note that (50) has a non-vacant antecedent since $U_{i+1} \cap \mathcal{Q}_j \neq \emptyset$ by definition of J_{i+1}, this allows using Farkas' lemma below. Let $\rho_{i+1}(\mathbf{x}) = \vec{a}\cdot\mathbf{x} + a_0$ and $f_i(\mathbf{x}) = \vec{b}\cdot\mathbf{x} + b_0$, where $\vec{a}$ and $\vec{b}$ are row vectors of n elements each. Recall that $\mathcal{Q}_j$ is given as a system of inequalities $A''_j\mathbf{x}'' \le \mathbf{c}''_j$, where A''_j is a matrix of dimension $m \times 2n$. Using these representations for ρ_{i+1}, f_i, and $\mathcal{Q}_j$ we can present (50) as follows:

$$\frac{\begin{pmatrix} A''_j \\ \vec{b},\ -\vec{b} \end{pmatrix} \cdot \mathbf{x}'' \le \begin{pmatrix} \mathbf{c}''_j \\ 0 \end{pmatrix}}{(-\vec{a},\ \ \vec{a}) \cdot \mathbf{x}'' \le 0}$$

Farkas' Lemma guarantees the existence of a vector $\vec{\mu}_j = (\mu_{j1}, \ldots, \mu_{jm}) \ge 0$, and a scalar $\xi_j \ge 0$, such that

$$-\vec{\mu}_j \cdot A''_j + \xi_j \cdot \left(-\vec{b},\ \ \vec{b}\right) = (\vec{a},\ \ -\vec{a})\,, \tag{51}$$

$$\vec{\mu}_j \cdot \mathbf{c}''_j \le 0. \tag{52}$$

This means that

$$\left(\vec{a} + \xi_j\cdot\vec{b},\ \ -(\vec{a} + \xi_j\cdot\vec{b})\right) = -\vec{\mu}_j \cdot A''_j\,. \tag{53}$$

Now since the entries of $\vec{\mu}_j$ are non-negative, from $A''_j\mathbf{x}'' \le \mathbf{c}''_j$ we get $\vec{\mu}_j\cdot A''_j\mathbf{x}'' \le \vec{\mu}_j\cdot\mathbf{c}''_j \le 0$. By (53),

$$-\vec{\mu}_j \cdot A''_j\mathbf{x}'' = \left(\vec{a} + \xi_j\cdot\vec{b},\ \ -(\vec{a} + \xi_j\cdot\vec{b})\right) \cdot \mathbf{x}'' = (\vec{a} + \xi_j\cdot\vec{b}) \cdot (\mathbf{x} - \mathbf{x}'),$$

so we get

$$\forall \mathbf{x}'' \in \mathcal{Q}_j\ .\ (\vec{a} + \xi_j\cdot\vec{b}) \cdot (\mathbf{x} - \mathbf{x}') \ge 0. \tag{54}$$

Note that $\xi_j\cdot\vec{b}\cdot(\mathbf{x} - \mathbf{x}') = \xi_j\cdot\Delta f_i(\mathbf{x}'')$, and that by ($47_i$,$48_i$) we have $\Delta f_i(\mathbf{x}'') \ge 0$ over $\overline{U}_i$, and thus over $\mathcal{Q}_j \subseteq \overline{U}_{i+1} \subseteq \overline{U}_i$. This means that (54) still holds when replacing ξ_j by any $\xi \ge \xi_j$. Now define $\xi = \max\{\xi_j \mid j \in J_{i+1}\}$, then (54) holds for any $j \in J_{i+1}$ and this ξ. Since $\overline{U}_{i+1} = \bigcup_{j\in J_{i+1}} \mathcal{Q}_j$, we get

$$\forall \mathbf{x}'' \in \overline{U}_{i+1}\ .\ (\vec{a} + \xi\cdot\vec{b}) \cdot (\mathbf{x} - \mathbf{x}') \ge 0\,. \tag{55}$$

Now we show that $f_{i+1}(\mathbf{x}) = \rho_{i+1}(\mathbf{x}) + (\xi + 1)\cdot f_i(\mathbf{x})$ satisfies (46_{i+1}–48_{i+1}).

(46_{i+1}) By (46_i) we know that $f_i(\mathbf{x}) \ge 0$ over $U_i \supset U_{i+1}$, and by (43_i) we know that $\rho_{i+1}(\mathbf{x}) \ge 0$ over U_{i+1}. Thus, for any $\mathbf{x}'' \in U_{i+1}$ we have $f_{i+1}(\mathbf{x}) = \rho_{i+1}(\mathbf{x}) + (\xi + 1)\cdot f_i(\mathbf{x}) \ge \rho_{i+1}(\mathbf{x}) \ge 0$.

(47_{i+1}) Pick an arbitrary $\mathbf{x}'' \in \overline{U}_{i+1} \setminus U_{i+2}$, and consider the two complementary cases $\mathbf{x}'' \in U_{i+1} \setminus U_{i+2}$ and $\mathbf{x}'' \notin U_{i+1} \setminus U_{i+2}$:

- (*a*) If $\mathbf{x}'' \in U_{i+1}\setminus U_{i+2} \subseteq U_{i+1}$, then by ($48_i$) we get $\Delta f_i(\mathbf{x}'') = 0$ and by (44_{i+1}) we get $\Delta\rho_{i+1}(\mathbf{x}'') > 0$. Thus, $\Delta f_{i+1}(\mathbf{x}'') = \Delta\rho_{i+1}(\mathbf{x}'') + (\xi + 1)\cdot\Delta f_i(\mathbf{x}'') = \Delta\rho_{i+1}(\mathbf{x}'') > 0$;
- (*b*) If $\mathbf{x}'' \notin U_{i+1}\setminus U_{i+2}$, then $\mathbf{x}'' \in (\overline{U}_{i+1}\setminus U_{i+1})\setminus U_{i+2} = \overline{U}_{i+1}\setminus U_{i+1}$. Write $\Delta f_{i+1}(\mathbf{x}'')$ as $(\vec{a}+\xi\cdot\vec{b})\cdot(\mathbf{x}-\mathbf{x}')+\Delta f_i(\mathbf{x}'')$. On one hand $\mathbf{x}'' \in \overline{U}_{i+1}\setminus U_{i+1} \subseteq \overline{U}_{i+1}$ so by (55) we get $(\vec{a}+\xi\cdot\vec{b})\cdot(\mathbf{x}-\mathbf{x}') \ge 0$, and on the other hand $\mathbf{x}'' \in \overline{U}_{i+1}\setminus U_{i+1} \subseteq \overline{U}_i\setminus U_{i+1}$ so by (47_i) we get $\Delta f_i(\mathbf{x}'') > 0$. Thus $\Delta f_{i+1}(\mathbf{x}'') = (\vec{a}+\xi\cdot\vec{b})\cdot(\mathbf{x}-\mathbf{x}')+\Delta f_i(\mathbf{x}'') \ge \Delta f_i(\mathbf{x}'') > 0$.

(48_{i+1}) Pick an arbitrary $\mathbf{x}'' \in U_{i+2}$. By (45_{i+1}) we have $\Delta\rho_{i+1}(\mathbf{x}'') = 0$, and by ($48_i$), since $U_{i+2} \subset U_{i+1}$, we have $\Delta f_i(\mathbf{x}'') = 0$. Thus,

$$\Delta f_{i+1}(\mathbf{x}'') = \Delta\rho_{i+1}(\mathbf{x}'') + (\xi + 1)\cdot\Delta f_i(\mathbf{x}'') = 0 + (\xi + 1)\cdot 0 = 0\,.$$

This completes the proof. □

Example 5.30. We compute f_1, f_2 and f_3 for the weak *LRF* $\tau = \langle x_1, x_2 - x_1, x_3\rangle$ of Example 5.25. So we have

$$\rho_1(x_1,x_2,x_3) = x_1, \quad \rho_2(x_1,x_2,x_3) = x_2 - x_1, \quad \rho_3(x_1,x_2,x_3) = x_3.$$

We let $A''_i\mathbf{x}'' \le \mathbf{c}''_i$, for $1 \le i \le 3$, be the constraint representations of the transition polyhedra.

(f_1) We set $f_1(x_1,x_2,x_3) = \rho_1(x_1,x_2,x_3) = x_1$, as in the base-case of the induction.

(f_2) We have $\overline{U}_2 = \mathcal{Q}_2 \cup \mathcal{Q}_3$, thus we solve (51,52) twice, once with $A''_2\mathbf{x}'' \le \mathbf{c}''_2$ and once with $A''_3\mathbf{x}'' \le \mathbf{c}''_3$. In both cases

$$(\vec{a}, -\vec{a}) = (-1,1,0,1,-1,0), \qquad (-\vec{b}, \vec{b}) = (-1,0,0,1,0,0).$$

We get $\xi_1 = 0$ and $\xi_2 = 1$, and thus we take $\xi = 1$. Then we define

$$f_2(x_1,x_2,x_3) = \rho_2(x_1,x_2,x_3) + (\xi+1)\cdot f_1(x_1,x_2,x_3) = x_2 + x_1.$$

(f_3) We have $\overline{U}_3 = \mathcal{Q}_3$, thus we solve (51,52) for $A''_3\mathbf{x}'' \le \mathbf{c}''_3$, $(\vec{a}, -\vec{a}) = (0,0,1,0,0,-1)$ and $(-\vec{b}, \vec{b}) = (-1,1,0,1,-1,0)$. We get $\xi = 0$, and thus

$$f_3(x_1,x_2,x_3) = \rho_3(x_1,x_2,x_3) + (\xi+1)\cdot f_2(x_1,x_2,x_3) = x_3 + x_2 + x_1.$$

Now we show how to use $f_1,\dots,f_d$ of Lemma 5.29 in order to construct a *LLRF* for $\mathcal{Q}_1,\dots,\mathcal{Q}_k$. We first state an auxiliary definition.

Definition 5.31. For affine functions $\rho_1,\dots,\rho_j : \mathbb{Q}^n \to \mathbb{Q}$, and positive constants $\delta_1,\dots,\delta_j$, define $\mathbf{R}(\langle\rho_1,\dots,\rho_j\rangle, \langle\delta_1,\dots,\delta_j\rangle)$ to be the set of $\mathbf{x}'' \in \mathbb{Q}^{2n}$ for which there is an $1 \le i \le j$ satisfying (37–39). We say that such transitions $\mathbf{x}''$ are ranked by $\langle\rho_1,\dots,\rho_j\rangle$ (with $\delta_1,\dots,\delta_j$), or, to name the position, that they are ranked by ρ_i in $\mathbf{R}(\langle\rho_1,\dots,\rho_j\rangle, \langle\delta_1,\dots,\delta_j\rangle)$.

In the next lemma we construct a *LLRF* τ_ℓ that ranks all transitions of $\mathcal{Q}_\ell$, for each $1 \le \ell \le k$. Afterwards, we show how $\tau_1,\dots,\tau_k$ are combined into a *LLRF* τ for $\mathcal{Q}_1,\dots,\mathcal{Q}_k$.

LEMMA 5.32. *Let $1 \le d' \le d$ be the largest d' such that $U_{d'} \cap \mathcal{Q}_\ell \ne \emptyset$ for a given $\mathcal{Q}_\ell$. Then, $\tau_\ell = \langle\rho'_1,\dots,\rho'_{d'}\rangle$, where $\rho'_i = f_i + i - 1$, is a LLRF for $\mathcal{Q}_\ell$.*

PROOF. For $1 \le i \le d'$, let $X_i = U_i \cap \mathcal{Q}_\ell$. Note that $X_1,\dots,X_{d'}$ are closed polyhedra, $\mathcal{Q}_\ell = X_1 \supseteq \ldots \supseteq X_{d'} \ne \emptyset$, and $X_{d'} \cap U_{d'+1} = \emptyset$. We find $\delta_1,\dots,\delta_{d'}$ such that

$$\mathbf{R}(\langle\rho'_1,\dots,\rho'_{d'}\rangle, \langle\delta_1,\dots,\delta_{d'}\rangle) \supseteq X_1\,. \tag{56}$$

This implies the lemma's statement since $X_1 = \mathcal{Q}_\ell$. The proof is by induction, where we start from $i = d'$ and proceed backwards. In the i-th step we find δ_i such that

$$R_i \overset{\text{def}}{=} \mathbf{R}(\langle f_i^{[i]},\dots,f_{d'}^{[i]}\rangle, \langle i\cdot\delta_i, i\cdot\delta_{i+1},\dots,i\cdot\delta_{d'}\rangle) \supseteq X_i\,, \tag{57_i}$$

where $f_j^{[i]} = f_j + j - i$. Then, for $i = 1$ we get (56). First note that $\Delta f_j^{[i_1]} = \Delta f_j^{[i_2]} = \Delta f_j$ for any $1 \le i_1 < i_2 \le d'$, this relation is fundamental to our proof. The intuition behind the offset $j - i$ in $f_j^{[i]}$ is explained below, at the beginning of the induction step.

Base-case. We take $i = d'$, then $f_{d'}^{[d']} = f_{d'}$ and thus $R_{d'} = \mathbf{R}(\langle f_{d'} \rangle, \langle d' \cdot \delta_{d'} \rangle)$. Since $X_{d'} \subseteq U_{d'}$ and $X_{d'} \cap U_{d'+1} = \emptyset$, then, for any $\mathbf{x}'' \in X_{d'}$, by ($46_{d'}$) we have $f_{d'}(\mathbf{x}) \geq 0$ and by ($47_{d'}$) we have $\Delta f_{d'}(\mathbf{x}'') > 0$. Now since $X_{d'}$ is a closed polyhedron and $\Delta f_{d'}$ is positive over $X_{d'}$, $\Delta f_{d'}$ must have a minimum $\mu > 0$ in $X_{d'}$. Define $\delta_{d'} = \frac{\mu}{d'}$, then $\Delta f_{d'}(\mathbf{x}'') \geq \mu = d' \cdot \delta_{d'}$. Thus, $X_{d'} \subseteq R_{d'}$.

Induction hypothesis. $X_{i+1} \subseteq R_{i+1}$.

Induction step. We find a value for δ_i, and show that $X_i \subseteq R_i$. Note that R_i uses the same $\delta_{i+1}, \ldots, \delta_{d'}$ as R_{i+1}.

Let us first intuitively explain how the induction step is carried out. We first split X_i into two sets, C_i and $X_i \setminus C_i$, and then show that each transition in $X_i \setminus C_i$ is ranked by $f_j^{[i]}$ for some $j > i$, and that each transition in C_i is ranked by $f_i^{[i]}$. To construct C_i, we simply start by considering the set of transitions that violate the $LLRF$ conditions (37-39) for all components $j > i$. This set is not closed, and, in order close it, we include also transitions that are on the "edge" (simply by turning strict inequalities to non-strict ones). Being closed is fundamental for a later step in the proof. Going back to the definition of $f_i^{[j]}$, the reason for which we use the offset $j - i$ (so it becomes larger as i becomes smaller) can be explained as moving ranked transitions away from some "edge". Next we define C_i, and then prove the desired properties of $X_i \setminus C_i$ and C_i.

Recall that C_i should be a superset of the transitions that are not ranked by any component $i \leq j \leq d'$ in R_i. Note that for any $i \leq j \leq d'$, by (47_j,48_j) we have $\Delta f_j^{[i]}(\mathbf{x}'') = \Delta f_j \geq 0$ for any $\mathbf{x}'' \in X_i$, thus it is not possible to violate (37). This means that if $\mathbf{x}''$ is not ranked by some $i < j \leq d'$ in R_i, then one of the following must hold:

— $\Delta f_j^{[i]}(\mathbf{x}'') < i \cdot \delta_j$ for any $i < j \leq d'$, to violate (39); or
— if there is $i < j' \leq d'$ for which $\Delta f_{j'}^{[i]}(\mathbf{x}'') \geq i \cdot \delta_{j'}$, assuming it is the smallest j', then there must be $l \leq j'$ for which $f_l^{[i]}(\mathbf{x}) < 0$, to violate (38).

The set of transitions that satisfy either of the above conditions is not necessarily closed — due to the use of strict inequalities. To obtain a closed set, we simply turn $<$ to $\leq$, and define C_i to be the set of all transitions $\mathbf{x}'' \in X_i$ for which one of the following holds

$$\forall i < j \leq d' \;.\; \Delta f_j^{[i]}(\mathbf{x}'') \leq i \cdot \delta_j \;, \tag{58}$$

$$\exists l \geq i \;.\; (\forall i < j < l \;.\; \Delta f_j^{[i]}(\mathbf{x}'') \leq i \cdot \delta_j) \wedge f_l^{[i]}(\mathbf{x}) \leq 0 \;. \tag{59}$$

Thus C_i is closed, and consists of a finite union of closed polyhedra.

We now prove that each transition in $X_i \setminus C_i$ is ranked by $f_j^{[i]}$, for some $i < j \leq d'$, in R_i. Pick an arbitrary transition $\mathbf{x}'' \in X_i \setminus C_i$, we show that it is ranked by $f_j^{[i]}$ in R_i, for some $j > i$. To see this, note the following:

— Since $\mathbf{x}'' \notin C_i$, it violates (58) and (59). To violate (58), there must be $i < j \leq d'$ for which

$$\Delta f_j^{[i]}(\mathbf{x}'') > i \cdot \delta_j \;. \tag{60}$$

Take minimal such j, then, for any $i < j' < j$, we have $\Delta f_{j'}^{[i]}(\mathbf{x}'') \leq i \cdot \delta_{j'}$. This means that the first conjunct of (59) is not violated by $\mathbf{x}''$ for any $i < l \leq j$, and thus, to violate (59), the second conjunct $f_l^{[i]}(\mathrm{x}) \leq 0$ must be violated, that is:

$$\forall i < l \leq j \ . \ f_l^{[i]}(\mathrm{x}) > 0 \,. \tag{61}$$

— Let $i \leq l \leq d'$. Since $X_l = U_l \cap \mathcal{Q}_\ell$ is not empty, $\mathcal{Q}_\ell \subseteq \overline{U}_l$. This means that $\mathrm{x}'' \in \overline{U}_l$, and thus by ($47_l$,$48_l$) we have

$$\Delta f_l^{[i]}(\mathrm{x}'') = \Delta f_l(\mathrm{x}'') \geq 0 \,. \tag{62}$$

Moreover, since $\mathrm{x}'' \in X_i \subseteq U_i$, by ($46_i$) we have

$$f_i^{[i]}(\mathrm{x}) = f_i(\mathrm{x}) \geq 0 \tag{63}$$

Inequalities (60–63) show that x'' is ranked by $f_j^{[i]}$ in R_i.

Now we show that the transitions of C_i are ranked by $f_i^{[i]}$ in R_i, for some δ_i. If $C_i = \emptyset$ then we simply take $\delta_i = \delta_{i+1}$, and clearly $X_i \subseteq R_i$ (since the transitions of $X_i \setminus C_i$ are ranked as we have seen above independently from δ_i). Assume $C_i \neq \emptyset$. We first claim that $C_i \cap X_{i+1} = \emptyset$. To see this, take $\mathrm{x}'' \in X_{i+1}$, by the induction hypothesis we have $X_{i+1} \subseteq R_{i+1}$ and thus there must be $f_j^{[i+1]}$, for some $i < j \leq d'$, that ranks x'', thus:

— $\Delta f_j^{[i]}(\mathrm{x}'') = \Delta f_j^{[i+1]}(\mathrm{x}'') \geq (i+1)\cdot\delta_j > i\cdot\delta_j$, so (58) is violated;
— $f_l^{[i+1]}(\mathrm{x}) \geq 0$ for any $i < l \leq j$, and thus $f_l^{[i]}(\mathrm{x}) = f_l^{[i+1]}(\mathrm{x}) + 1 \geq 1$. This means that (59) cannot be true for any $i < l \leq j$, it also cannot be true for any $j < l \leq d'$ since $\Delta f_j^{[i]}(\mathrm{x}'') > i\cdot\delta_j$ as we have seen in the previous point.

Now since $C_i \cap X_{i+1} = \emptyset$ and $C_i \subseteq X_i$ we get $C_i \subseteq X_i \setminus X_{i+1}$. We also know that $X_i \setminus X_{i+1} \subseteq \overline{U}_i \setminus U_{i+1}$ by definition, and that by (47_i) we have $\Delta f_i(\mathrm{x}'') > 0$ throughout $\overline{U}_i \setminus U_{i+1}$. This means that $\Delta f_i(\mathrm{x}'') > 0$ throughout C_i as well. Now since C_i is a finite union of closed polyhedra, $\Delta f_i(\mathrm{x}'')$ must have a minimum $\mu > 0$. Define $\delta_i = \frac{\mu}{i}$ then $f_i^{[i]}(\mathrm{x}'') = f_i(\mathrm{x})'' \geq \mu = i\cdot\frac{\mu}{i}$. Moreover, by ($46_i$) we have $f_i(\mathrm{x}) \geq 0$ and thus $f_i^{[i]}(\mathrm{x}) = f_i(\mathrm{x}) \geq 0$. This proves that $\mathrm{x}'' \in C_i$ is ranked by $f_i^{[i]}$ in R_i. □

LEMMA 5.33. *$\tau = \langle \rho'_1, \ldots, \rho'_d \rangle$, where $\rho'_j = f_j + j - 1$, is a LLRF for $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$. Moreover, it has a minimal dimension, at most n.*

PROOF. That τ is a *LLRF* follows immediately from Lemma 5.32, because the transitions of each $\mathcal{Q}_\ell$ are ranked in τ_ℓ, and each τ_ℓ is a prefix of τ. The minimality of the dimension follows from that of the weak *LLRF*: if there were a shorter *LLRF*, since every *LLRF* is a weak *LLRF*, it would contradict Lemma 5.26. □

Example 5.34. Consider again the weak *LLRF* of Example 5.25, and $f_1 = x_1$, $f_2 = x_2+x_1$ and $f_3 = x_3+x_2+x_1$ that we have computed in Example 5.30. The corresponding *LLRF* is $\tau = \langle x_1, x_1 + x_2 + 1, x_1 + x_2 + x_3 + 2 \rangle$, with $\delta_1 = 1$, $\delta_2 = \frac{1}{2}$ and $\delta_3 = \frac{1}{3}$.

THEOREM 5.35. LEXLINRF($\mathbb{Q}$) *is PTIME-decidable.*

PROOF. Procedure LLRFSYN, which has polynomial-time complexity by Lemma 5.13, is complete for the existence of a weak *LLRF*. If no weak *LLRF* exists then no *LLRF* exists either, and by Lemma 5.33, if one exists then there is a *LLRF*. □

Note that if only termination is of interest, then there is no reason to actually perform the construction of Lemmas 5.29 and 5.32, it suffices to check the existence of a weak *LLRF*. Ranking functions are also used to bound the number of iterations of loops, as discussed in the next subsection. In this context, an explicit upper bound is desirable, so we may need to carry out the construction of Lemmas 5.29 and 5.32. This can be done in polynomial time since computing $f_1, \ldots, f_n$ as in Lemma 5.29 only requires solving k LP problems of bit-size polynomially bounded in the input bit-size.

5.4. Lexicographic Ranking Functions and Iteration Bounds

Alias et al. [2010] showed how lexicographic ranking functions can be used to bound the number of steps in a program—in our restricted form of programs this is just the number of iterations of the loop. What is sought is a symbolic bound, as an expression in terms of the input variables. $LRFs$ clearly provide linear bounds, and $LLRFs$ provide polynomial bounds when each component of the $LLRF$ has a linear upper bound (derived using a linear-invariant generator). Clearly, this bound is at most the product of the bounds on the individual components, and hence a polynomial of degree given by the dimension of the $LLRF$ (this motivates the interest in $LLRFs$ of minimal dimension). In the next theorem we show that, in fact, for SLC loops we can always find a piecewise linear bound (this observation applies whether one is interested in ranking all rational points or just integer ones). Note that Alias et al. [2010] proved that an SLC loop has a $LLRF$ if and only if it has a LRF, and thus has a linear bound on the number of iterations. However, our definition of $LLRF$ captures some SLC loops that do not have a LRF, as seen in Example 2.12.

THEOREM 5.36. *Let $\mathcal{Q}$ be the transition polyhedron of an SLC loop, $\langle\rho_1, \ldots, \rho_d\rangle$ a (weak) $LLRF$ inferred by Procedure* LLRFSYN, *and $\tau = \langle\rho'_1, \ldots, \rho'_{d'}\rangle$ a $LLRF$ as constructed in Lemma 5.32 with corresponding $\delta_1, \ldots, \delta_{d'}$. Given an input $\mathbf{x} \in \mathbb{Q}^n$, let j be the minimum $1 \le j \le d'$ such that $\rho'_j(\mathbf{x}) < 0$, or $j = d'$ if no one exists, then $\sum_{i=1}^{j-1}(\lfloor\rho'_i(\mathbf{x})/\delta_i\rfloor + 1)$ is an upper bound on the number of iterations of $\mathcal{Q}$ when starting from* $\mathbf{x}$.

PROOF. By Lemma 5.29, any $\mathbf{z}'' \in \mathcal{Q}$ satisfies $\Delta\rho'_i(\mathbf{z}'') \ge 0$; for any $1 \le i \le d'$, which means that once the i-th component of τ become negative, it is then disabled and cannot rank any transition anymore (since it remains negative). In addition, when a transition is ranked by the i-th component, $\Delta\rho'_i(\mathbf{x}'') \ge \delta_i$ which, together with the above argument, means that the i-th component of τ can rank at most $\lfloor\rho'_i(\mathbf{x})/\delta_i\rfloor + 1$ transitions before it becomes negative. Now since every transition in the execution trace must be ranked by some component ρ'_i of τ, and i cannot be $\ge j$ since such components are disabled right from the beginning, we get the upper bound $\sum_{i=1}^{j-1}(\lfloor\rho'_i(\mathbf{x})/\delta_i\rfloor + 1)$. $\square$

Remarks:

(1) If we are only interested in an upper bound up to a constant factor, we can avoid the construction of Lemmas 5.29 and 5.32 because $\sum_{i=1}^{j-1}(\lfloor\rho'_i(\mathbf{x})/\delta_i\rfloor + 1)$ is $O(\sum_{i=1}^{d} \max(0, \rho_i(\mathbf{x}))$.
(2) The theorem is easily extended to conclude that the piecewise linear bound is also valid for MLC loops, when ρ_d ranks at least one transition from each $\mathcal{Q}_i$, that is, $U_d \cap \mathcal{Q}_i \neq \emptyset$ for all $1 \le k \le d$.

One of the interesting parts of [Alias et al. 2010] is the way they compute an iteration bound which is sometimes better than the product of the bounds on the $LLRF$ components. The idea: Since ρ always decreases, the number of steps is bounded by the number of distinct values it takes throughout the computation. Let $\mathcal{C}$ be the polyhedron which circumscribes the state space (in our case, the loop condition); $\rho(\mathcal{C})$ is a d-dimensional polyhedron, and, assuming that the program computes over integers, the number of steps is bounded by the number of integer points in this polyhedron, i.e., $|I(\rho(\mathcal{C}))|$. Alias et al. estimate this number using techniques related to Ehrhart

Algorithm 2: Find a point in the relative interior

```
InteriorPoint($\mathcal{S}$)
Input: Space of quasi-$LRFs$ $\mathcal{S}$
Output: A point $(\lambda_0, \lambda)$ in the relative interior
begin
1    for $i = 1 \to n$ do
2        $a \leftarrow$ minimize $\lambda_i$ wrt $\mathcal{S}$
3        $b \leftarrow$ maximize $\lambda_i$ wrt $\mathcal{S}$
4        if $a = b$ then $c_i = a$

6        else pick $c_i$ in the non-closed interval $(a, b)$, prioritizing $0$ and integers

8        $\mathcal{S} \leftarrow \mathcal{S} \wedge \{\lambda_i = c_i\}$
9    $c_0 \leftarrow$ minimize $\lambda_0$ wrt $\mathcal{S}$
10   return $(c_0,\vec{c})$
```

polynomials, as implemented in the PolyLib library [Wilde 1993]. Such an approach can also be used with our class of functions, but it is an open problem how to get the best results out of such computations. For example, is it possible to find a computation method that will always get a piecewise linear bound in the situations described by the above theorem?

6. PROTOTYPE IMPLEMENTATION

The different algorithms presented in this paper for synthesizing $LRFs$ an $LLRFs$, both for the general cases and the special PTIME cases, have been implemented. Our tool, iRANKFINDER, can be tried out via http://www.loopkiller.com/irankfinder. It receives as input an MLC loop in constraint representation, and allows applying different algorithms for LINRF($\mathbb{Z}$), LINRF($\mathbb{Q}$), LEXLINRF($\mathbb{Z}$), or LEXLINRF($\mathbb{Q}$). For $LRFs$, the implementation includes the algorithms of Theorems 3.19 and 4.24. By default it uses the second one since the first one relies on the generator representation of the transition polyhedron, which may take exponential time to compute. For $LLRFs$ it uses Algorithm 1.

Our algorithm for synthesizing non-trivial quasi-$LRFs$, as described in Lemma 5.4, requires finding a point in the relative interior of a polyhedron $\mathcal{S}$. Note that $\mathcal{S}$ is of dimension $n' = n + 1 + \sum_{i=1}^{k} 2m_i$ and is defined by $m' = k(8n + 2) + \sum_{i=1}^{k} 2m_i$ inequalities, where m_i is the number of inequalities in $\mathcal{Q}_i$. Existing algorithms for finding an interior point require solving at most n' or m' LP problems, and they have polynomial-time complexity [Fukuda 2013, Sec. 8.3]. Now note that instead of finding a point in the relative interior of $\mathcal{S}$, we could also project $\mathcal{S}$ onto $\vec{\lambda}$, and then find a point in the relative interior of the resulting polyhedron $\mathcal{S}_{|\vec{\lambda}}$. It is easy to see that Lemma 5.4 remains valid. In our implementation we find such point without actually computing $\mathcal{S}_{|\vec{\lambda}}$, by solving only $2n + 1$ LP problems. The underlying procedure is depicted in Algorithm 2, it finds values for $\vec{\lambda}$ iteratively as follows: in the i-th iteration it computes the minimum and maximum values of λ_i in $\mathcal{S}$, and then sets λ_i to a value that lies between those extremes. Once all λ_i are computed, we look for the minimum compatible value of λ_0, and then $(c_0,\vec{c})$ is the desired point. We do not claim that the complexity of this algorithm is polynomial, since we add $\lambda_i = c$ to $\mathcal{S}$ in each iteration and thus the bit-size might grow exponentially. However, we have experimentally observed that it performs far better than an algorithm that finds a point in the relative interior of $\mathcal{S}$. Note that at Line 7, we prioritize 0 over any other coefficient, as a heuristic to obtain "small" ranking functions. Moreover, we prioritize integer over fractional coefficients. Both measures are intended to get more readable results, but we think they may also improve time bounds inferred from our ranking functions.

Computing the integer hull of a polyhedron, in the case of LINRF($\mathbb{Z}$) and LEXLINRF($\mathbb{Z}$), is done by first decomposing its set of inequalities into independent components, and then computing the integer hull of each component separately. Each set of inequalities is first matched against the PTIME cases of sections 4.1. If this matching fails, the integer hull is computed using the algorithm described by Charles et al. [2009]. Note that this algorithm supports only bounded polyhedra, the integer hull of an unbounded polyhedron is computed by considering a corresponding bounded one [Schrijver 1986, 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 Section 4.3, when this option is used, completeness of LINRF($\mathbb{Z}$) is not guaranteed.

The Parma Polyhedra Library [Bagnara et al. 2008b] is used for converting between generator and constraints representations, solving (mixed) LP problems, etc.

7. RELATED WORK

There are several works [Sohn and Gelder 1991; Colón and Sipma 2001; Podelski and Rybalchenko 2004a; Mesnard and Serebrenik 2008; Alias et al. 2010] 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 (7) and (8) of Definition 2.9 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. [2005c] and Cook et al. [2010] that directly address the LINRF($\mathbb{Z}$) problem for MLC loops. Below, we comment in more detail on each of these works.

Sohn and Gelder [1991] considered MLC loops with variables ranging over $\mathbb{N}$. These are abstractions of loops from logic programs. The loops were relaxed from $\mathbb{N}$ to $\mathbb{Q}_+$ before seeking a LRF, however, this is not explicitly mentioned. The main observation in this work is that the duality theorem of LP [Schrijver 1986, 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 [1990] 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 [2008] formally proved that the techniques of Sohn and Gelder [1991] 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 [2004a]. 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 [Schrijver 1986, 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 [2004]. One of the reasons for the impact of this work is its use in the Terminator tool [Cook et al. 2006], which demonstrated the use of *LRFs* in termination analysis of complex, real-world programs.

Bagnara et al. [2012] proved that the methods of Mesnard and Serebrenik [2008] and Podelski and Rybalchenko [2004a] 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 [2001], in the context of termination analysis, and by Feautrier [1992a], in the context of automatic parallelization of computations. Colón and Sipma [2001] 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 Podelski and Rybalchenko [2004a]. Completeness was claimed, however it was not explicitly mentioned that the variables range over $\mathbb{Q}$ and not $\mathbb{Z}$ (the programs in the examples used integer variables). In this work the input loop comes with an initial condition on the input, which is used to infer a supporting invariant.

Feautrier [1992a] 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. Feautrier [1992b] extends this work to lexicographic rankings; this work was subsequently extended by Alias et al. [2010] to *LLRF* generation, as described below.

Cook et al. [2010] observed that the Farkas-lemma based solution is complete for $\text{LinRF}(\mathbb{Z})$ 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 $\text{LinRF}(\mathbb{Z})$. In fact, the main issue in that work is the synthesis of ranking functions for machine-level integers (bit-victors).

Bradley et al. [2005c] directly addressed the $\text{LinRF}(\mathbb{Z})$ problem for *MLC* loops, and stated that the methods of Colón and Sipma [2001] and Podelski and Rybalchenko [2004a] are not complete for $\text{LinRF}(\mathbb{Z})$. 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 ranking function coefficients. Our Corollary 3.22 shows that it is possible to deterministically bound this depth, that turns their algorithm into a complete one, though still exponential. In addition to *LRFs*, this technique is extended for inferring linear invariants over $\mathbb{Z}$.

The interest of Bradley et al. [2005c] was in *MLC* loops in which *integer division by constants* is allowed. It is incorrect to replace integer division $x' = \frac{x}{c}$ by precise division, but the operation can be simulated by two paths of linear constraints: $\{x \geq$

$0, c \cdot x' + y = x, 0 \leq y \leq c-1\}$ and $\{x \leq 0, c \cdot x' - y = x, 0 \leq y \leq c-1\}$. This illustrates the usefulness of (multipath) linear-constraint loops.

Codish et al. [2005] 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 if and only if 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 if and only if 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 the constraint $x'_i \geq x'_j + c$ if and only if it includes $x_i \geq x_j + c$. They showed how to balance the loop while preserving its termination behavior. Recently, Bozga et al. [2012] presented similar results for SLC loops defined by octagonal relations, implying that *termination* is decidable (even PTIME) for such loops.

Cousot [2005] 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.

The earliest work that we know, that addresses lexicographic-linear ranking functions, is that of Colón and Sipma [2002]. As in their previous work, they use LP methods based on the computation of polars. The $LLRF$ is not constructed explicitly but can be inferred from the results of the algorithm. Bradley et al. [2005a] employed a constraint-solving approach to search for lexicographic-linear ranking functions, where a template solution is set up and linear programming is used to find the unknown coefficients in the template. Bradley et al. [2005b] also relaxed the notion of ranking functions to functions that *eventually* decrease, while in another work [Bradley et al. 2005d] they considered MLC loops with polynomial transitions and the synthesis of lexicographic-polynomial ranking functions. All these works actually tackle an even more complex problem, since they also search for *supporting invariants*, based on the transition constraints and on given preconditions. Harris et al. [2011] demonstrate that it is advantageous, to a tool that is based on a CEGAR loop, to search for $LLRFs$ instead of constructing transition invariants from $LRFs$ only as in the original Terminator tool. They use a simplified version of the template method of Bradley et al. [2005a]. Similar observations have been reported by Cook et al. [2013], Brockschmidt et al. [2013] and Larraz et al. [2013].

Alias et al. [2010] again extended the Farkas-lemma based solution for LINRF($\mathbb{Q}$) to the construction of $LLRFs$. Like Colón and Sipma [2002], they do it for programs with an arbitrary control-flow graph. Unlike the latter, they prove completeness of their procedure (which means completeness over the rationals), and their algorithm is of polynomial time. The goal of Alias et al. [2010] 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. Their construction produces a function of minimum dimension (within their class of ranking functions, which is narrower than ours, as discussed in Section 2).

Decidability and complexity of termination (in general, not necessarily with $LRFs$ or $LLRFs$) of SLC and MLC loops has been intensively studied for different classes of constraints. For SLC loops, Tiwari [2004] proved that the problem is decidable when the update is affine linear and the variables range over $\mathbb{R}$. Braverman [2006] proved that this holds also for $\mathbb{Q}$, and for the homogeneous case it holds for $\mathbb{Z}$. Both considered universal termination, i.e., for all input. Also, in both cases they allow the use of strict inequalities in the condition. Ben-Amram et al. [2012] showed that the termination of

SLC loops is undecidable if the use of a single irrational coefficient is allowed, as well as for MLC loops with at least two paths, and certain other variants.

For some specific forms of integer MLC loops termination is decidable: Extending previous work on Size-Change Termination [Lee et al. 2001], Ben-Amram [2011] proved that termination is decidable (more precisely: PSPACE-complete) for MLC loops with monotonicity constraints (as defined above). Bozzelli and Pinchinat [2012] further extended the result (still PSPACE-complete) for Gap Constraints, which are constraints of the form $X - Y \geq c$ where $c \in \mathbb{N}$ and X and Y are variables or primed variables. This is, clearly, an extension of monotonicity constraints, which in particular allows for more precise representation of relations of variables to constants. Ben-Amram [2008] proved that for difference constraints over the integers, specifically updates of the form $x_i - x'_j \geq c$ where $c \in \mathbb{Z}$, and guards $x_i \geq 0$, the termination problem becomes undecidable. However for a subclass in which each target (primed) variable might be constrained only once (in each path of a multiple-path loop) the problem is PSPACE-complete.

Regarding ranking functions, Ben-Amram [2011] shows that every terminating program of the considered form has a ranking function which is *piecewise lexicographic*. This is achieved by transforming the program (by splitting CFG nodes) into one that is guaranteed to have a $LLRF$. Such a result is probably achievable for the gap constraints of Bozzelli and Pinchinat [2012] as well. However, it is unknown how to explicitly construct ranking functions for difference constraints as those used by Ben-Amram [2008].

8. CONCLUDING REMARKS

We have studied the Linear Ranking problem for SLC and MLC 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 SLC 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 $x = \frac{10}{3}$. Note that this loop does not have a LRF over the integers.

We have obtained results similar to the above regarding the LEXLINRF($\mathbb{Z}$) problem, the existence of lexicographic-linear ranking functions. Our polynomial-time algorithm for LEXLINRF($\mathbb{Q}$) is also new, and extends the class of functions that can be found by the previously known polynomial-time algorithm of Alias et al. [2010]. Our algorithm is optimal, in the sense that it synthesizes $LLRFs$ with minimal dimension.

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) lexicographic-linear (or linear) function τ_ν with each node ν, so that on a transition from ν to ν' we should have $\tau_\nu(\mathrm{x}) \succ_{lex} \tau_{\nu'}(\mathrm{x}')$. 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 [Mesnard and Serebrenik 2008; Alias et al. 2010]. The considerations regarding the complexity of the corresponding problems 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 Example 3.6 and Loop (1) of Section 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 Section 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.

In Section 2.2 we have discussed the differences between our $LLRFs$ and those of Alias et al. [2010] and Bradley et al. [2005a]. This raises the question of how our results extend to these other definitions of $LLRFs$. Alias et al. [2010] already show that their algorithm is complete and PTIME over the rationals, and it is easy to show that it is complete over the integers when computing the integer hull first, in which case our special PTIME case also apply. Over the integers, the decision problem is clearly coNP-hard (using the same reduction of Section 3.1), and we conjuncture that it is in coNP as well. The algorithm of Bradley et al. [2005a] is exponential over the rationals, since they search also for supporting invariants starting from a given preconditions. If one is interested only in $LLRFs$ which are valid for any input, we conjuncture that it can be done in polynomial time, by iteratively seeking functions that are similar to our quasi-$LRFs$. Over the integers, the corresponding decision problem is clearly coNP-hard (using the same reduction of Section 3.1), and we conjuncture that it is in coNP as well. The technical development of the above conjunctures is left for future work.

In Section 4.3 we have discussed the LinRF($\mathbb{Z}$) problem for loops specified by octagonal relations. We showed that it is not possible to obtain a polynomial-time algorithm that is based on computing the integer hull as in our special PTIME cases. The question of whether this special case of LinRF($\mathbb{Z}$) is in PTIME or not is still open.

In this paper we have considered $LRFs$ and $LLRFs$ which are valid for *any* initial input. However, loops often come with a precondition that restricts the space of valid input. This is the case, for example, of the counter-example "lassos" generated by approaches that are based on CEGAR [Cook et al. 2006; Cook et al. 2013; Brockschmidt et al. 2013; Harris et al. 2011]. The complexity classification of the corresponding decision problems, both over rationals and integers, is still open. Recent work [Heizmann et al. 2013; Leike 2013] provides partial answers for the rational case.

A more general definition for $LLRFs$ can by obtained by requiring (10) of Definition 2.11 to hold only for $j = i$. This is similar to the definition of Bradley et al. [2005a], however, it is more general since it does not require a fixed association of ranking-function components with the paths of the loop. Additional generalizations of linear ranking functions are *eventual ranking functions* [Bagnara and Mesnard 2013] and Polyranking functions [Bradley et al. 2005b]. The complexity classification of the corresponding decision problems, over the integers (and in the latter case, also over rationals), is still open.

Regarding the potential practical impact of our results, recent work [Cook et al. 2013; Brockschmidt et al. 2013] argues that the performance of a Terminator-like [Cook et al. 2006] tool can be dramatically improved by the use of $LLRFs$, instead of disjunctive well-founded relations [Podelski and Rybalchenko 2004b]. This is demonstrated by their experiments, despite of using an exponential-time algorithm. While we have not implemented our methods in a complete tool, their results indicate that using a polynomial-time $LLRFs$ algorithm could significantly improve such analyzers. In addition, our special PTIME cases that are based on affine linear updates are also appealing in practice, because loops (in real programs) that operate on integer variables often have this form. Thus, for such cases, one can trust the answer of the polynomial-time algorithm over the rationals.

Our algorithm for computing $LLRFs$, similarly to others [Alias et al. 2010; Larraz et al. 2013], is based on iteratively eliminating transitions. When the algorithm fails to find a $LLRF$, it is guaranteed that no infinite execution can involve any of the eliminated transitions infinitely often. In other words, any infinite execution must have a suffix that consists only of the remaining transitions (the potentially non-terminating kernel). Ganty and Genaim [2013] show how this kernel can be used to infer preconditions on the input that guarantee termination, however, their technique is developed for a more general kind of termination witness, namely disjunctive well-founded relations [Podelski and Rybalchenko 2004b]. Exploiting this approach in our setting might have practical advantages, since the performance bottleneck in the algorithm of Ganty and Genaim [2013] is the computation of the potentially non-terminating kernel, which we can compute (or approximate) in polynomial time.

Finally, a theoretical study does not capture all aspects of the relative merits of different types of termination witnesses. In practice, first, the performance of algorithms is a more involved issue than just a complexity class; e.g., some polynomial algorithms are better than others, and some super-polynomial algorithms are nonetheless practical. In addition, considerations such as simplicity of the termination witnesses, information provided for certifying the witness, etc., may be important, depending on the application. Thus, we conclude that empirical studies and algorithm-engineering are still an important objective for future research.

REFERENCES

Elvira Albert, Puri Arenas, Samir Genaim, and Germán Puebla. 2011. Closed-Form Upper Bounds in Static Cost Analysis. *J. Autom. Reasoning* 46, 2 (2011), 161–203.

Elvira Albert, Puri Arenas, Samir Genaim, German Puebla, and Damiano Zanardini. 2007. COSTA: Design and Implementation of a Cost and Termination Analyzer for Java Bytecode. In *Formal Methods for Components and Objects, FMCO'07 (LNCS)*, Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever (Eds.), Vol. 5382. Springer, 113–132.

Christophe Alias, Alain Darte, Paul Feautrier, and Laure Gonnord. 2010. Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs. In *Static Analysis Symposium, SAS'10 (LNCS)*, Radhia Cousot and Matthieu Martel (Eds.), Vol. 6337. Springer, 117–133.

Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. 2008a. An Improved Tight Closure Algorithm for Integer Octagonal Constraints. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'08 (LNCS)*, Francesco Logozzo, Doron Peled, and Lenore D. Zuck (Eds.), Vol. 4905. Springer, 8–21.

Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. 2008b. 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 (2008), 3–21.

Roberto Bagnara and Fred Mesnard. 2013. Eventual Linear Ranking Functions. In *Proceedings of the 15th International Symposium on Principles and Practice of Declarative Programming, PPDP 2013*. ACM Press, 229–238.

Roberto Bagnara, Fred Mesnard, Andrea Pescetti, and Enea Zaffanella. 2012. A new look at the automatic synthesis of linear ranking functions. *Inf. Comput.* 215 (2012), 47–67.

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

Amir M. Ben-Amram. 2011. Monotonicity Constraints for Termination in the Integer Domain. *Logical Methods in Computer Science* 7, 3 (2011).

Amir M. Ben-Amram and Samir Genaim. 2013. On the Linear Ranking Problem for Integer Linear-Constraint Loops. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '13)*. ACM, New York, NY, USA, 51–62.

Amir M. Ben-Amram, Samir Genaim, and Abu Naser Masud. 2012. On the Termination of Integer Loops. *ACM Trans. Program. Lang. Syst.* 34, 4, Article 16 (Dec. 2012), 24 pages.

Marius Bozga, Radu Iosif, and Filip Konecný. 2012. Deciding Conditional Termination. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'12 (LNCS)*, Cormac Flanagan and Barbara König (Eds.), Vol. 7214. Springer, 252–266.

Laura Bozzelli and Sophie Pinchinat. 2012. Verification of Gap-Order Constraint Abstractions of Counter Systems. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'12 (LNCS)*, Viktor Kuncak and Andrey Rybalchenko (Eds.), Vol. 7148. Springer, 88–103.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2005a. Linear Ranking with Reachability. In *Computer Aided Verification, CAV'05 (LNCS)*, Kousha Etessami and Sriram K. Rajamani (Eds.), Vol. 3576. Springer, 491–504.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2005b. The Polyranking Principle. In *International Colloquium on Automata, Languages and Programming, ICALP'05 (LNCS)*, Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung (Eds.), Vol. 3580. Springer, 1349–1361.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2005c. Termination Analysis of Integer Linear Loops. In *Concurrency Theory, CONCUR 2005 (LNCS)*, Martín Abadi and Luca de Alfaro (Eds.), Vol. 3653. Springer, 488–502.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2005d. Termination of Polynomial Programs. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'05 (LNCS)*, Radhia Cousot (Ed.), Vol. 3385. Springer, 113–129.

Mark Braverman. 2006. Termination of Integer Linear Programs. In *Computer Aided Verification, CAV'06 (LNCS)*, Thomas Ball and Robert B. Jones (Eds.), Vol. 4144. Springer, 372–385.

Marc Brockschmidt, Byron Cook, and Carsten Fuhs. 2013. Better Termination Proving through Cooperation. In *Computer Aided Verification, CAV 2013 (Lecture Notes in Computer Science)*, Natasha Sharygina and Helmut Veith (Eds.), Vol. 8044. Springer, 413–429.

Maurice Bruynooghe, Michael Codish, John P. Gallagher, Samir Genaim, and Wim Vanhoof. 2007. Termination analysis of logic programs through combination of type-based norms. *ACM Trans. Program. Lang. Syst.* 29, 2 (2007).

Philip J. Charles, Jacob M. Howe, and Andy King. 2009. Integer Polyhedra for Program Analysis. In *Algorithmic Aspects in Information and Management, AAIM'09 (LNCS)*, Andrew V. Goldberg and Yunhong Zhou (Eds.), Vol. 5564. Springer, 85–99.

Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. 2005. Testing for Termination with Monotonicity Constraints. In *International Conference on Logic Programming, ICLP'05 (LNCS)*, Maurizio Gabbrielli and Gopal Gupta (Eds.), Vol. 3668. Springer, 326–340.

Michael Colón and Henny Sipma. 2001. Synthesis of Linear Ranking Functions. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'01 (LNCS)*, Tiziana Margaria and Wang Yi (Eds.), Vol. 2031. Springer, 67–81.

Michael Colón and Henny Sipma. 2002. Practical Methods for Proving Program Termination. In *Computer Aided Verification, 14th International Conference, CAV'02,Copenhagen, Denmark, July 27-31, 2002, Proceedings (LNCS)*, Ed Brinksma and Kim Guldstrand Larsen (Eds.), Vol. 2404. Springer, 442–454.

Byron Cook, Daniel Kroening, Philipp Rümmer, and Christoph M. Wintersteiger. 2010. Ranking Function Synthesis for Bit-Vector Relations. In *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS'10 (LNCS)*, Javier Esparza and Rupak Majumdar (Eds.), Vol. 6015. Springer, 236–250.

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. 2006. Termination proofs for systems code. In *Programming Language Design and Implementation, PLDI'06*, Michael I. Schwartzbach and Thomas Ball (Eds.). ACM, 415–426.

Byron Cook, Abigail See, and Florian Zuleger. 2013. Ramsey vs. Lexicographic Termination Proving. In *Tools and Algorithms for the Construction and Analysis of Systems,TACAS 2013 (Lecture Notes in Computer Science)*, Nir Piterman and Scott A. Smolka (Eds.), Vol. 7795. Springer, 47–61.

Patrick Cousot. 2005. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'05 (LNCS)*, Radhia Cousot (Ed.), Vol. 3385. 1–24.

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

Paul Feautrier. 1992a. Some efficient solutions to the affine scheduling problem. I. One-dimensional time. *International Journal of Parallel Programming* 21, 5 (1992), 313–347.

Paul Feautrier. 1992b. Some efficient solutions to the affine scheduling problem. II. Multidimensional time. *International Journal of Parallel Programming* 21, 6 (1992), 389–420.

Komei Fukuda. 2013. Lecture: Polyhedral Computation, Spring 2013. Available at http://www-oldurls.inf.ethz.ch/personal/fukudak/lect/pclect/notes2013. (February 2013).

Pierre Ganty and Samir Genaim. 2013. Proving Termination Starting from the End. In *Proceedings of the 25th International Conference on Computer Aided Verification, CAV 2013 (Lecture Notes in Computer Science)*, Natasha Sharygina and Helmut Veith (Eds.), Vol. 8044. Springer, 397–412.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability*. W.H. Freeman and Co., New York.

Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. 2004. Automated Termination Proofs with AProVE. In *Rewriting Techniques and Applications, RTA'04 (LNCS)*, Vincent van Oostrom (Ed.), Vol. 3091. Springer, 210–220.

William R Harris, Akash Lal, Aditya V Nori, and Sriram K Rajamani. 2011. Alternation for termination. In *Static Analysis Symposium, SAS 2011*. LNCS, Vol. 6337. Springer, 304–319.

Mark E. Hartmann. 1988. *Cutting Planes and the Complexity of the Integer Hull*. Ph.D. Dissertation. School of Operations Research and Industrial Engineering, Cornell University.

Warwick Harvey. 1999. Computing Two-Dimensional Integer Hulls. *SIAM J. Comput.* 28, 6 (1999), 2285–2299.

Warwick Harvey and Peter J. Stuckey. 1997. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Australasian Computer Science Conference, ACSC'97*. 102–111.

Matthias Heizmann, Jochen Hoenicke, Jan Leike, and Andreas Podelski. 2013. Linear Ranking for Linear Lasso Programs. In *Automated Technology for Verification and Analysis*, Dang Hung and Mizuhito Ogawa (Eds.). Lecture Notes in Computer Science, Vol. 8172. Springer International Publishing, 365–380. DOI:http://dx.doi.org/10.1007/978-3-319-02444-8_26

R. M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.). Plenum Press, New York, 85–103.

Richard M. Karp and Christos H. Papadimitriou. 1980. On Linear Characterizations of Combinatorial Optimization Problems. In *Symp. on Foundations of Computer Science, FOCS'80*. IEEE Computer Society, 1–9.

Daniel Kroening, Natasha Sharygina, Aliaksei Tsitovich, and Christoph Wintersteiger. 2010. Termination Analysis with Compositional Transition Invariants. In *Computer Aided Verification, CAV 2010 (LNCS)*, Vol. 6174. Springer, 89–103.

Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. 2013. Proving termination of imperative programs using Max-SMT. In *Formal Methods in Computer-Aided Design, FMCAD 2013*. IEEE, 218–225.

Jean-Louis Lassez. 1990. Querying Constraints. In *Symposium on Principles of Database Systems*. ACM Press, 288–298.

Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. 2001. The size-change principle for program termination. In *Symposium on Principles of Programming Languages, POPL'01*, Chris Hankin and Dave Schmidt (Eds.). 81–92.

Jan Leike. 2013. *Ranking Function Synthesis for Linear Lasso Programs*. Master's thesis. University of Freiburg, Department of Computer Science.

Naomi Lindenstrauss and Yehoshua Sagiv. 1997. Automatic Termination Analysis of Prolog Programs. In *International Conference on Logic Programming, ICLP'97*, Lee Naish (Ed.). MIT Press, 64–77.

Stephen Magill, Ming-Hsien Tsai, Peter Lee, and Yih-Kuen Tsay. 2010. Automatic numeric abstractions for heap-manipulating programs. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010*, Manuel V. Hermenegildo and Jens Palsberg (Eds.). ACM, 211–222.

Frédéric Mesnard and Alexander Serebrenik. 2008. Recurrence with affine level mappings is P-time decidable for CLP(R). *TPLP* 8, 1 (2008), 111–119.

Antoine Miné. 2006. The octagon abstract domain. *Higher-Order and Symbolic Computation* 19, 1 (March 2006), 31–100.

Andreas Podelski and Andrey Rybalchenko. 2004a. A Complete Method for the Synthesis of Linear Ranking Functions. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'04 (LNCS)*, Bernhard Steffen and Giorgio Levi (Eds.), Vol. 2937. Springer, 239–251.

Andreas Podelski and Andrey Rybalchenko. 2004b. Transition Invariants. In *19th IEEE Symposium on Logic in Computer Science, LICS 2004*. IEEE Computer Society, 32–41.

Peter Z. Revesz. 2009. Tightened Transitive Closure of Integer Addition Constraints. In *Symposium on Abstraction, Reformulation, and Approximation, SARA'09*, Vadim Bulitko and J. Christopher Beck (Eds.).

Andrey Rybalchenko. 2004. *Temporal Verification with Transition Invariants*. Ph.D. Dissertation. Universität des Saarlandes.

Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York.

Kirack Sohn and Allen Van Gelder. 1991. Termination Detection in Logic Programs using Argument Sizes. In *Symposium on Principles of Database Systems*, Daniel J. Rosenkrantz (Ed.). ACM Press, 216–226.

Fausto Spoto, Fred Mesnard, and Étienne Payet. 2010. A termination analyzer for Java bytecode based on path-length. *ACM Trans. Program. Lang. Syst.* 32, 3 (2010).

Éva Tardos. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* 34 (1986), 250–256.

Ashish Tiwari. 2004. Termination of Linear Programs. In *Computer Aided Verification, CAV'04*, Rajeev Alur and Doron Peled (Eds.). LNCS, Vol. 3114. Springer, 387–390.

Doran K. Wilde. 1993. *A library for doing polyhedral operations*. Technical Report PI 785. IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires), France.