Breaking the Mold: Nonlinear Ranking Function Synthesis Without Templates

Shaowei Zhu(✉) and Zachary Kincaid

Princeton University, Princeton, NJ 08540, USA
{shaoweiz,zkincaid}@cs.princeton.edu

Abstract. This paper studies the problem of synthesizing (lexicographic) polynomial ranking functions for loops that can be described in polynomial arithmetic over integers and reals. While the analogous ranking function synthesis problem for *linear* arithmetic is decidable, even checking whether a *given* function ranks an integer loop is undecidable in the nonlinear setting. We side-step the decidability barrier by working within the theory of linear integer/real rings (LIRR) rather than the standard model of arithmetic. We develop a termination analysis that is guaranteed to succeed if a loop (expressed as a formula) admits a (lexicographic) polynomial ranking function. In contrast to template-based ranking function synthesis in *real* arithmetic, our completeness result holds for lexicographic ranking functions of unbounded dimension and degree, and effectively subsumes linear lexicographic ranking function synthesis for linear *integer* loops.

Keywords: termination · ranking functions · polynomial ranking functions · lexicographic ranking functions · monotone · nonlinear arithmetic

1 Introduction

Ranking function synthesis refers to the problem of finding a well-founded metric that decreases at each iteration of a loop. It is a critical subroutine in modern termination analyzers like Terminator [12], Ultimate Automizer [16], and COMPACT [26]. One could synthesize ranking functions via a *template*, i.e., fixing a particular form of ranking functions to be considered while leaving parameters as free variables, and encoding the conditions for the function to rank the given loop as a logical formula, thereby reducing the synthesis problem to a constraint-solving problem. Provided that the resulting constraint-solving problem is decidable, this method yields a complete procedure for synthesizing ranking functions that match the template. In particular, the template-based method is the basis of complete synthesis of ranking functions for linear and lexicographic linear ranking functions for loops whose bodies and guards can be expressed in linear real or integer arithmetic [3,23]. A limitation of the approach is that it is only complete with respect to template languages that can be defined by finitely

A. Gurfinkel and V. Ganesh (Eds.): CAV 2024, LNCS 14681, pp. 431–452, 2024.
https://doi.org/10.1007/978-3-031-65627-9_21

many parameters (e.g., we may define a template for all linear terms or degree-2 polynomials, but not polynomials of unbounded degree).[1]

In this paper, we study the problem of synthesizing polynomial ranking functions for nonlinear loops. There are two apparent obstacles. The first obstacle results from the difficulty of reasoning about nonlinear arithmetic. Nonlinear *integer* arithmetic is undecidable, and so even checking whether a *given* function ranks a loop is undecidable, let alone synthesizing one. While nonlinear real arithmetic is decidable, it has high complexity–prior work has explored *incomplete* constraint-solving approaches to avoid the cost of decision procedures for real arithmetic [1,13], but this sacrifices the completeness property typically enjoyed by template-based methods. The second obstacle is that the set of all polynomials cannot be described as a template language with finitely many parameters, thus precluding complete ranking function synthesis based on the template method.

To tackle the undecidability problem, we adopt a weak theory of nonlinear arithmetic **LIRR** that is decidable [18]. For the infinite template problem, we first compute the finite set of polynomials that are entailed to be *bounded* modulo **LIRR** by the loop, and use them to define a template language with finitely many parameters to describe "candidate terms" for ranking functions. We then show that synthesis of ranking functions consisting of non-negative linear combinations of these candidate terms can be reduced to a constraint-solving problem in linear arithmetic. The adoption of **LIRR** ensures that we do not lose completeness in any of the above steps, i.e., any ranking function modulo **LIRR** can be written as a nonnegative combination of the "candidate terms" in the template. We thus have a procedure for synthesizing polynomial ranking functions that is sound for the reals, and *complete* in the sense that if a polynomial ranking function exists for a formula (modulo **LIRR**), then the analysis will find it. Furthermore, we extend this analysis to one that is sound for the integers and complete relative to lexicographic polynomial ranking functions (modulo **LIRR**).

Using the framework of algebraic termination analysis [26], we extend our termination analysis on loops (represents as formulas) to whole programs (including nested loops, recursive procedures, etc.). The completeness of the proposed procedures leads to *monotone* end-to-end termination analyses for whole programs. Informally, monotonicity guarantees that if the analysis can prove termination of a program P and P is transformed to a program P' in a way that provides more information about its behavior (e.g., by decorating the program with invariants discovered by an abstract interpreter) then the analysis is certain to prove termination of P' as well.

Our experimental evaluation establishes that the procedure based on polynomial ranking function and lexicographic polynomial ranking function synthesis

[1] One may imagine using the template paradigm to search for polynomial ranking functions of successively higher degree until one is found; however, this yields a complete *semi-algorithm*, which fails to terminate if no polynomial ranking function exists.

with the background theory of **LIRR** is competitive for SV-COMP termination benchmarks, especially for the nonlinear programs.

2 Background

Linear Algebra and Polyhedral Theory

In the following, we use **linear space** to mean a linear space over the field of rationals $\mathbb{Q}$. Let L be a linear space. A set $C \subseteq L$ is **convex** if for every $p, q \in C$ and every $\lambda \in [0,1]$, we have $\lambda p + (1-\lambda)q \in C$. We use $conv(S)$ to denote the **convex hull** of a set $S \subseteq L$, which is the smallest convex set that contains S. A set Q is a **polytope** if it is the convex hull of a finite set. A set $C \subseteq L$ is a **(convex) cone** if it contains 0 and is closed under addition and multiplication by $\mathbb{Q}^{\geq 0}$ (nonnegative rationals). For a set $G \subseteq L$, its **conical hull** is the smallest cone that contains G, defined as $cone(G) = \{\lambda_1 g_1 + \cdots + \lambda_m g_m : \lambda_i \in \mathbb{Q}^{\geq 0}, g_i \in G\}$. Given any $A, B \subseteq L$, we use $A + B \triangleq \{a + b : a \in A, b \in B\}$ to denote their Minkowski sum.

A set $P \subseteq L$ is a **polyhedron** if $P = cone(R) + conv(V)$, where R, V are finite sets in L, and use the notation $P = V\text{-}rep(R, V)$. Convex polyhedra are effectively closed under intersection; that is, there is a procedure `intersect` such that for any finite $R_1, V_1, R_2, V_2 \subseteq L$ we have

$$V\text{-}rep(\texttt{intersect}\,(R_1, V_1, R_2, V_2)) = V\text{-}rep(R_1, V_1) \cap V\text{-}rep(R_2, V_2)\ .$$

The Ring of Rational Polynomials

For a finite set of variables X, we use $\mathbb{Q}[X]$ to denote the ring of polynomials over X with rational coefficients, and $\mathbb{Q}[X]^1$ to denote the set of linear polynomials over X. A set $I \subseteq \mathbb{Q}[X]$ is an **ideal** if it contains zero, is closed under addition, and for every $p \in \mathbb{Q}[X]$ and $q \in I$ we have $pq \in I$. For a finite set $G = \{g_1, \ldots, g_n\} \subseteq \mathbb{Q}[X]$, we use $\langle G \rangle \triangleq \{p_1 g_1 + \cdots + p_n g_n : p_1, \ldots, p_n \in \mathbb{Q}[X]\}$ to denote the ideal generated by the elements in G. By Hilbert's basis theorem, we have that every ideal in $\mathbb{Q}[X]$ can be written as $\langle G \rangle$ for some finite set G. Equivalently, for any ascending chain of ideals $I_1 \subseteq I_2 \subseteq \ldots$ in $\mathbb{Q}[X]$, there exists an index j such that $I_j = I_k$ for all $k \geq j$.

Note that $\mathbb{Q}[X]$ is a linear space over $\mathbb{Q}$, and so cones, polytopes, and polyhedra consisting of polynomials are defined as above. We say that a cone $C \subseteq \mathbb{Q}[X]$ is **algebraic** if it is the Minkowski sum of an ideal and a finitely-generated convex cone [18]. For finite sets of polynomials $Z, P \subseteq \mathbb{Q}[X]$, we use

$$alg.cone_X(Z, P) \triangleq \left\{ \sum_{z \in Z} q_z z + \sum_{p \in P} \lambda_p p : q_z \in \mathbb{Q}[X], \lambda_p \in \mathbb{Q}^{\geq 0} \right\}$$

to denote the algebraic cone generated by Z and P; we call Z and P the "zeros" and "positives" of the cone, respectively. When the set of variables is clear, we often omit the subscript and just write $alg.cone(Z, P)$.

For any algebraic cone $C \subseteq \mathbb{Q}[X]$, the set of *linear* polynomials in C forms a convex polyhedron. We use `linearize` to denote the operation that computes this set—that is, for any finite $Z, P \subseteq \mathbb{Q}[X]$, we have

$$V\text{-}rep(\texttt{linearize}\,(Z,P)) = alg.cone\,(Z,P) \cap \mathbb{Q}[X]^1 \ .$$

There is a procedure `inverse-hom` for computing the inverse image of an algebraic cone under a ring homomorphism ([18], Theorem 9). More precisely, let $alg.cone_X(Z,P)$ be an algebraic cone, Y be a set of variables, and $f : \mathbb{Q}[Y] \to \mathbb{Q}[X]$ be a ring homomorphism, then

$$alg.cone_Y(\texttt{inverse-hom}\,(Z,P,f,Y)) = \{p \in \mathbb{Q}[Y] : f(p) \in alg.cone_X(Z,P)\} \ .$$

In this paper it will be useful to define a common generalization algebraic cones and convex polyhedra, which we call a *algebraic polyhedra*. We say that a set of polynomials $R \subseteq \mathbb{Q}[X]$ is an **algebraic polyhedron** if it is the Minkowski sum of an algebraic cone and a convex polytope[2]. An algebraic polyhedron can be represented by a triple $\langle Z, P, V\rangle$ where Z, P, V are finite sets of polynomials; such a triple represents the algebraic polyhedron

$$alg.polyhedron\,(Z,P,V) \triangleq alg.cone\,(Z,P) + conv\,(V) \ .$$

The Arithmetic Theory LIRR and Consequence Finding

We use the following syntax for formulas:

$$F, G \in \textbf{Formula} ::= p \leq q \mid p = q \mid Int(p) \mid F \wedge G \mid F \vee G \mid \neg F$$

where p and q denote polynomials with rational coefficients over some set of variable symbols. We regard the reals $\mathbb{R}$ as the standard interpretation of this language, with *Int* identifying the subset of integers $\mathbb{Z} \subset \mathbb{R}$.

Kincaid et al. [18] defined another class of interpretations for the above language of formulas called *linear integer/real rings*. A linear integer/real ring is a commutative ring equipped with an order and an integer predicate which obeys certain axioms of the theories of linear real and linear integer arithmetic. The standard interpretation $\mathbb{R}$ is an example of a linear integer/real ring. A "nonstandard" example is the ring $\mathbb{Q}[x]$, where $p \leq q$ iff p precedes q lexicographically (e.g., $-x^3 < x < x^2 - x < x^2 < x^2 + x$) and $Int(p)$ holds iff p's coefficients are integers.

The fact that the theory **LIRR** of linear integer/real rings (refer to [18] for an axiomatization) admits such nonstandard (and inequivalent) models means that the *theory* is incomplete. Nevertheless it has desirable algorithmic properties that we will make use of in our ranking function synthesis procedures. We discuss the limitations brought by **LIRR** in Example 3.

[2] Recalling that a *convex* polyhedron is the Minkowski sum of a *finitely generated convex* cone and a polytope.

Since the reals $\mathbb{R}$ is a model for **LIRR**, if we have $F \models_{\mathbf{LIRR}} G$, we also have $F \models_{\mathbb{R}} G$. However, in this paper we are mostly concerned with entailment modulo **LIRR** rather than the standard model, thus we abbreviate $F \models_{\mathbf{LIRR}} G$ to $F \models G$ by default.

For a formula F and a set of variables X, we use

$$\mathbf{C}_X(F) \triangleq \{p \in \mathbb{Q}[X] : F \models p \geq 0\}$$

to denote the **nonnegative cone** of F (over X). For example, given $X = \{x, y\}$

$$\mathbf{C}_X(x = 2 \wedge y \leq 1) = \mathit{alg.cone}(\{x - 2\}, \{1, 1 - y\}) \ .$$

$\mathbf{C}_X(F)$ is an algebraic cone, and there is an algorithm for computing it (Algorithm 2 of [18]), which we denote by $\texttt{consequence}(F, X)$. We furthermore have that if $\langle Z, P\rangle = \texttt{consequence}(F, X)$, then $\langle Z\rangle = \{z \in \mathbb{Q}[X] : F \models z = 0\}$.

Transition Systems and Transition Formulas

For a set of variables X, we use $X' \triangleq \{x' : x \in X\}$ denote a set of "primed copies". For a polynomial $p \in \mathbb{Q}[X]$, we use p' to denote the polynomial in $\mathbb{Q}[X']$ obtained by replacing each variable x with its primed copy x'. A **transition formula** over a set of variables X is a formula F whose free variables range over X and X'. We use $\mathbf{TF}(X)$ to denote the set of all transition formulas over X. For a transition formula $F \in \mathbf{TF}(X)$ and real valuation $v, v' \in \mathbb{R}^X$, we use $v \rightarrow_F v'$ to denote that $\mathbb{R}, [v, v'] \models F$, where $\mathbb{R}$ denotes the standard model and $[v, v']$ denotes the valuation that maps each $x \in X$ to $v(x)$ and each $x' \in X'$ to $v'(x)$. A **real execution** of a transition formula F is an infinite sequence $v_0, v_1, \cdots \in \mathbb{R}^X$ such that for each i, we have $v_i \rightarrow_F v_{i+1}$; we say that $v_0, v_1, \ldots$ is an **integer execution** if additionally each $v_i \in \mathbb{Z}^X$. We say that F *terminates over* $\mathbb{R}$ if it has no real executions, and F *terminates over* $\mathbb{Z}$ if it has no integer executions.

Ranking Functions

Let $F \in \mathbf{TF}(X)$ be a transition formula. We say that $r \in \mathbb{Q}[X]$ is a **polynomial ranking function** (**PRF**) for F (modulo **LIRR**) if $F \models 0 \leq r$ and $F \models r' \leq r - 1$. The set of all polynomial ranking functions of F (modulo **LIRR**) is denoted $\mathbf{PRF}(F)$.

Lemma 1. *If* $\mathbf{PRF}(F) \neq \emptyset$*, then* F *terminates over* $\mathbb{R}$*.*

Proof. If $r \in \mathbf{PRF}(F)$, then $\lfloor r(X) \rfloor$ is a ranking function mapping $\mathbb{R}^X$ into $\mathbb{Z}$ that is well-ordered by a relation $\preceq$, defined as $x \preceq y$ iff $x \geq 0 \wedge x \leq y$, where $\leq$ is the usual order on the integers. □

We now consider lexicographic termination arguments. We define a *quasi-polynomial ranking function* (**QPRF**) for a transition formula $F \in \mathbf{TF}(X)$ (modulo **LIRR**) to be a polynomial $r \in \mathbb{Q}[X]$ such that

$$F \models r - r' \geq 0 \wedge r \geq 0 \ .$$

We say that a sequence of polynomials $r_1, \ldots, r_n \in \mathbb{Q}[X]$ is a dimension-n **weak lexicographic polynomial ranking function** (**WLPRF**) for F (modulo **LIRR**) if

$$\begin{aligned}
&r_1 \in \mathbf{QPRF}(F) \\
&r_2 \in \mathbf{QPRF}(F \wedge r_1' = r_1) \\
&\vdots \\
&r_n \in \mathbf{QPRF}\left(F \wedge \bigwedge_{i=1}^{n-1} r_i' = r_i\right) \\
&F \wedge \bigwedge_{i=1}^{n} r_i' = r_i \models \mathit{false} \ .
\end{aligned}$$

Lemma 3 sketches the proof that the existence of **WLPRF** proves termination of F over $\mathbb{Z}$.

Lemma 2. *Let $F \in \mathbf{TF}(X)$ be a transition formula. If $r \in \mathbb{Q}[X]$ is a quasi-ranking function for F, i.e., $F \models r' \leq r \wedge r \geq 0$, and furthermore $F \wedge r' = r$ terminates over the integers, then so does F.*

Proof. Since quasi-ranking functions are closed under scaling by nonnegative scalars, we may assume that r has integer coefficients without loss of generality. Suppose for a contradiction that F has an infinite integer execution $x_0, x_1, \ldots$. Since $r(x_i) \geq r(x_{i+1})$ for all i, and the range of r is restricted to $\mathbb{Z}^{\geq 0}$, there exists some n such that $r(x_n) = r(x_{n+1}) = \ldots$. But this is impossible since $F \wedge r' = r$ terminates over the integers. □

Lemma 3. *If a transition formula F admits a* **WLPRF** *(modulo the theory* **LIRR***), then F terminates over $\mathbb{Z}$.*

Proof. We prove this by induction on the dimension n of **WLPRF** of F. The base case holds vacuously when $n = 0$ since F is unsatisfiable, and the inductive case holds by Lemma 2. □

Note that Lemma 3 holds only for integer executions. Ben-Amram and Genaim [3] showed that existence of a weak lexicographic linear ranking function (LLRF) for a topologically closed linear formula implies existence of an LLRF for loop with real variables, but the argument fails for *nonlinear* formulas (even modulo **LIRR**). Consider the following **LIRR** transition formula over reals n, z

$$F \triangleq z \geq 0 \wedge n \geq 2 \wedge n' = 2n \wedge z \geq z' \wedge nz' = nz - 1 \ .$$

Then $F \models z \geq 0 \wedge z \geq z'$, and also $F \wedge z = z' \models \mathit{false}$. Thus z does decrease at every iteration of F and its value is bounded from below. However, F does not terminate since the rate at which z decreases diminishes too quickly.

3 Polynomial Ranking for LIRR Transition Formulas

In this section, we consider the problem of synthesizing polynomial ranking functions for transition formulas modulo **LIRR**. Observe that for a transition formula $F \in \mathbf{TF}(X)$, the polynomial ranking functions $\mathbf{PRF}(F)$ of F can be decomposed as $\mathbf{PRF}(F) = \mathit{Bounded}(F) \cap \mathit{Decreasing}(F)$ where $\mathit{Bounded}(F)$ are the bounded and decreasing polynomials of F, respectively:

$$\mathit{Bounded}(F) \triangleq \{p \in \mathbb{Q}[X] : F \models p \geq 0\}$$
$$\mathit{Decreasing}(F) \triangleq \{p \in \mathbb{Q}[X] : F \models p' \leq p - 1\}$$

Thus, one approach to computing $\mathbf{PRF}(F)$ is to compute the sets of bounded and decreasing polynomials, and then take the intersection.

First, we observe that we can use this strategy to synthesize *linear* ranking functions using the primitives defined in Sect. 2[3].

- The convex polyhedron of degree-1 polynomials of $\mathit{Bounded}(F)$ can be computed as $\texttt{linearize}(\texttt{consequence}(F, X))$,
- The convex polyhedron of degree-1 polynomials of $\mathit{Decreasing}(F)$ can be computed as follows. Define $f : \mathbb{Q}[X] \to \mathbb{Q}[X \cup X']$ to be the homomorphism mapping $x \mapsto x - x'$, and observe that

$$\mathit{Decreasing}(F) \cap \mathbb{Q}[X]^1 = \left\{p \in \mathbb{Q}[X]^1 : F \models f(p) - 1 \geq 0\right\}$$

  We proceed by first computing the polyhedron

$$Q \triangleq \left\{p + a : p \in \mathbb{Q}[X]^1, a \in \mathbb{Q}.F \models f(p) + a \geq 0\right\}$$

  as $\texttt{linearize}(\texttt{inverse-hom}(\texttt{consequence}(F, X \cup X'), f))$. Then we intersect Q with the hyperplane consisting of linear polynomials with constant coefficient -1, and then take the Minkowski sum with the singleton $\{1\}$ to get $\mathit{Decreasing}(F) \cap \mathbb{Q}[X]^1$.

The essential difficulty of adapting this strategy to find polynomial ranking functions of unbounded degree is that the function $g : \mathbb{Q}[X] \to \mathbb{Q}[X \cup X']$ mapping $p \mapsto p' - p$ is not a homomorphism (the function f defined above agrees with g on *linear* polynomials, but not on polynomials of greater degree).

Our method proceeds as follows. As we will later see in Algorithm 2, we can adapt the above strategy to compute the intersection of $\mathbf{PRF}(F)$ with some "template language" $\{a_1 p_1 + \cdots + a_n p_n : a_1, \ldots, a_n \in \mathbb{Q}\}$ for fixed polynomials

[3] This is essentially a recasting of the classic algorithms linear ranking function synthesis [3] for LRA, restated in our language.

$p_1, \ldots, p_n$. Our insight is to use the cone generators of *Bounded*(F) to define $p_1, \ldots, p_n$. This yields a ranking function synthesis procedure that, in general, is sound but incomplete; however, it is complete under the assumption that F is *zero-stable*. In Sect. 3.1 we define zero-stability and show that assuming zero-stability is essentially without loss of generality, and in Sect. 3 we define a procedure for computing $\mathbf{PRF}(F)$ for zero-stable F.

3.1 Zero-Stable Transition Formulas

Consider a transition formula F defined as

$$F \triangleq x = 0 \wedge y \geq 0 \wedge (x')^2 = y - y' - 1 \ .$$

Observe that $F \models x = 0$. F has a PRF $x^2 + y$, but it's hard to find in the sense that it's not a linear combination of the generators of *Bounded*(F) (x, $-x$, and y). But when $x' \neq 0$, the loop terminates immediately. Thus we can consider the restriction $F \wedge x' = 0$, which admits the linear ranking function y. The zero-stable restriction process we introduce below formalizes this process.

We define a transition formula $F \in \mathbf{TF}(X)$ to be **zero-stable** if for all polynomials $p \in \mathbb{Q}[X]$ such that $F \models p = 0$, it is the case that $F \models p' = 0$. We give an algorithm for computing the weakest zero-stable transition formula that entails the original formula in Algorithm 1, and we note that the algorithm preserves termination behavior (Lemma 4).

```
1 Function zero-stable-restrict(F)
      Input: A transition formula F ∈ TF(X).
      Output: The weakest zero-stable transition formula that entails F.
2     ⟨Z, _⟩ ←consequence(F,X);
      /* ⟨Z⟩ = {p ∈ ℚ[X] : F ⊨ p = 0}                                   */
3     repeat
4         Z' ← Z;
5         F ← F ∧ ⋀_{z∈Z} z' = 0;
6         ⟨Z, _⟩ ← consequence(F,X);
7     until ⟨Z⟩ = ⟨Z'⟩;
8     return F
```

Algorithm 1: The zero-stable restriction of a transition formula.

Lemma 4. *Let F be an* **LIRR** *transition formula, and*

$$\hat{F} \triangleq \texttt{zero-stable-restrict}(F) \ .$$

1. *Algorithm 1 computes the weakest zero-stable formula that entails F.*
2. *F terminates iff $\hat{F}$ terminates.*

Proof. Let $F^{(k)}$ and $Z^{(k)}$ denote the value of F and Z after the k-th iteration of the loop in Algorithm 1, respectively.

We first prove 1. Clearly, if Algorithm 1 terminates at some iteration n, then $\hat{F} = F^{(n)}$ is zero stable and entails F. It remains to show that (a) $F^{(n)}$ is the *weakest* such formula, and (b) the algorithm terminates.

(a) We show by induction that for any zero-stable formula G that entails F, it is the case that $G \models F^{(k)}$ for all k. The base case holds by assumption, since $G \models F = F^{(0)}$. Now suppose that $G \models F^{(k)}$, and we wish to show that $G \models F^{(k+1)}$. Since for each $z \in Z^{(k)}$ we have $G \models F^{(k)} \models z = 0$, and G is zero-stable, we know $G \models z' = 0$. It follows $G \models (F^{(k)} \wedge \bigwedge_{z \in Z^{(k)}} z' = 0) = F^{(k+1)}$.

(b) We show that Algorithm 1 terminates. Suppose that it does not. Then $\langle Z^{(0)} \rangle \subsetneq \langle Z^{(1)} \rangle \subsetneq \cdots$ forms an infinite strictly ascending chain of ideals of $\mathbb{Q}[X]$, contradicting Hilbert's basis theorem.

For 2, if F terminates then $\hat{F}$ clearly also terminates since $\hat{F} \models F$. To show that if $\hat{F}$ terminates then F must also terminate, we prove by induction that for any $k \geq 0$, $F^{(k+1)}$ terminates implies that $F^{(k)}$ terminates. We show this by arguing that any real execution of $F^{(k)}$ is also one of $F^{(k+1)}$. Let $v_0, v_1, \ldots$ be an execution of $F^{(k)}$. It is sufficient to show that $v_i \rightarrow_{F^{(k+1)}} v_{i+1}$ for all i. Since $v_{i+1} \rightarrow_{F^{(k)}} v_{i+2}$, we must have $z(v_{i+1}) = 0$ for all $z \in Z^{(k)}$, and so since $F^{(k+1)} = F^{(k)} \wedge \bigwedge_{z \in Z^{(k)}} z' = 0$ we have $v_i \rightarrow_{F^{(k+1)}} v_{i+1}$. □

Example 1. Consider running Algorithm 1 on a transition formula F

$$F : x = 0 \wedge y \geq 0 \wedge y' = -(x')^2 + y - 1 + z' \wedge z = x' \ .$$

In the first iteration of the loop, we discover a zero consequence x of F: $F \models x = 0$, and we then constrain the transition formula to be $F \wedge x' = 0$. Now since $F \models z = x'$, we get a new zero consequence z: $F \wedge x' = 0 \models z = 0$. We thus further constrain the transition formula to be $F \wedge x' = 0 \wedge z' = 0$. After adding these constraints, we can no longer find new zero consequences, and the resulting transition formula

$$F \wedge x' = 0 \wedge z' = 0 \equiv x = z = x' = z' = 0 \wedge y' = y - 1 \wedge y \geq 0$$

is zero-stable.

3.2 Complete Polynomial Ranking Function Synthesis

Assuming that a transition formula is zero-stable allows us to ignore polynomials in the ideal $\langle Z \rangle = \{p \in \mathbb{Q}[X] : F \models p = 0\}$ when synthesizing polynomial ranking functions, in the following sense. Suppose there exists $r \in \mathbf{PRF}(F)$ a polynomial ranking function for F, where $\langle Z, P \rangle = \texttt{consequence}\,(F, X)$. We can write r as $r = z + p$ with $z \in \langle Z \rangle$ and $p \in \mathit{cone}(P)$. Since F is zero-stable, we have $F \models p' \leq p - 1$, thus some polynomial in $\mathit{cone}(P)$ is decreasing. Thus, it is sufficient to search for decreasing polynomials in $\mathit{cone}(P)$. Algorithm 2 computes the complete set of **PRF** for zero-stable transition formulas, which is illustrated in the example below.

1 **Function** `prf-zero-stable`(F)
  Input: A zero-stable transition formula $F(X, X')$.
  Output: A tuple Z, R, V such that *alg.polyhedron*$(Z, R, V) = \mathbf{PRF}(F)$ of F.
2 $\langle Z, P\rangle \leftarrow$ `consequence`(F, X);
3 $Y \leftarrow \{y_p : p \in P\}$ be a set of fresh variables;
4 $f \leftarrow$ the homomorphism $\mathbb{Q}[Y] \to \mathbb{Q}[X]$ defined by $f(y_p) = p - p'$;
5 $\langle R', V'\rangle \leftarrow$ `linearize(inverse-hom(consequence(`F, $X \cup X'$`), `f, Y`))`;
   /* $\langle R_L, V_L\rangle$ *represents the polyhedron of linear terms with positive coefficients for variables and constant coefficient* -1. */
6 $R_L \leftarrow \{y : y \in Y\}, V_L \leftarrow \{-1\}$;
7 $\langle R_Y, V_Y\rangle \leftarrow$ `intersect(`R', V', R_L, V_L`)`;
   /* *Translate polyhedron from* $\mathbb{Q}[Y]^1$ *back to* $\mathbb{Q}[X]$, *and add constant* 1. */
8 $R \leftarrow \{r[y_p \mapsto p]_{p\in P} : r \in R_Y\}$;
9 $V \leftarrow \{1 + v[y_p \mapsto p]_{p\in P} : v \in V_Y\}$;
10 **return** $\langle Z, R, V\rangle$

Algorithm 2: Computing **PRF** for zero-stable transition formulas. We use notation $p[y \mapsto z_y : y \in S]$ to denote substitution of all variables $y \in S$ with z_y in a polynomial p.

Example 2. Consider running Algorithm 2 on a (zero-stable) transition formula

$$F: \quad nx \geq 0 \wedge n \geq 0 \wedge n' = n \wedge x \geq 0 \wedge z \geq 1 \\ \wedge ((z' = z - 1 \wedge x' = x) \vee (x' = x - 1 \wedge z' = z + n - 1)) \,.$$

The bounded polynomials of F (Line 1) is the algebraic cone defined by $Z = \{\emptyset\}$ and $P = \{nx, x, n, z - 1, 1\}$. Let $Y = \{t_{nx}, t_x, t_n, t_{z-1}, t_1\}$ be a fresh set of variables, let f be the ring homomorphism such that

$$\begin{aligned} f(t_{nx}) &= nx - n'x' \\ f(t_x) &= x - x' \\ f(t_n) &= n - n' \\ f(t_{z-1}) &= (z-1) - (z'-1) = z - z' \\ f(t_1) &= 1 - 1 = 0 \end{aligned}$$

After Line 5, we obtain the polyhedron of linear polynomials in the inverse image of the nonnegative cone of F under f, which is defined by the rays $R' = \{t_n, -t_n, t_1, -t_1, t_{nx} + t_{z-1} - 1, 1 - t_{nx} - t_{z-1}\}$ and one vertex $V' = \{0\}$. The subset of *V-rep*(R', V') of polynomials with nonnegative coefficients for variables and constant coefficient -1 (Line 7), is the polyhedron defined by rays $R_Y = \{t_{nx}, t_x, t_n, t_1\}$, and vertices $V_Y = \{t_{nx} + t_{z-1} - 1\}$. Finally, the algorithm returns the algebraic polyhedron with zeros $Z = \emptyset$, positives $R = \{nx, x, n, 1\}$, and vertices $V = \{nx + z\}$. Thus F has a non-empty set of polynomial ranking function modulo **LIRR** (e.g., it contains $nx + z$), and so we may conclude that F terminates over the reals.

We are then ready to prove the correctness of Algorithm 2.

Theorem 1 (Soundness and completeness of Algorithm 2). *For any zero-stable transition formula F*

$$\mathbf{PRF}(F) = \mathit{alg.polyhedron}(\texttt{prf-zero-inv}(F)) \ .$$

Proof. Let $Z, P, Y, f, R', V', R_L, V_L, R_Y, V_Y, R, V$ be as in Algorithm 2. Let $s : \mathbb{Q}[Y] \to \mathbb{Q}[X]$ be the homomorphism mapping $y_p \mapsto p$ (corresponding to the substitution on lines 8-9). Observe that for any linear combination of the Y variables $q = \sum_{p \in P} a_p y_p$, we have

$$f(q) = \sum_{p \in P} a_p f(y_p) = \sum_{p \in P} a_p (p - p') = \left(\sum_{p \in P} a_p p\right) - \left(\sum_{p \in P} a_p p'\right) = s(q) - s(q)' \ .$$

By definition (line 5), we have $\mathit{V\text{-}rep}(R_Y, V_Y) = \left\{q \in \mathbb{Q}[Y]^1 : F \models f(q) \geq 0\right\}$ and (line 6) $\mathit{V\text{-}rep}(R_L, V_L) = \mathit{cone}(Y) + \{-1\}$. It follows that the intersection of these two polyhedra (line 7) is

$$\mathit{V\text{-}rep}(R_Y, V_Y) = \{q - 1 : q \in \mathit{cone}(Y), F \models s(q) - s(q)' - 1 \geq 0\} \ .$$

Then by the construction of R and V (lines 7-8) we have

$$\mathit{V\text{-}rep}(R, V) = \{s(q) : q \in \mathit{cone}(Y), F \models s(q) - s(q)' - 1 \geq 0\} \ .$$

Letting $K = \mathit{V\text{-}rep}(R, V)$, we must show that $\mathbf{PRF}(F) = \langle Z \rangle + K$. We prove inclusion in both directions.

$\subseteq$ Let $r \in \mathbf{PRF}(F)$. Then we have $F \models r \geq 0$ and $F \models r - r' - 1 \geq 0$. Since $F \models r \geq 0$ and $\mathit{alg.cone}(Z, P) = \mathbf{C}_X(F)$, we have we have $r = z + p$ for some $z \in \langle Z \rangle$ and $p \in \mathit{cone}(P)$. To show $r = z + p \in \langle Z \rangle + K$, it is sufficient to show $p \in K$.
Write p as $\left(\sum_{t \in P} c_t t\right)$ for some $\{c_t\}_{t \in P} \subseteq \mathbb{Q}^{\geq 0}$. Let $q = \left(\sum_{t \in P} c_t y_t\right)$. Then we have $s(q) = p$ and $q \in \mathit{cone}(Y)$, so it is sufficient to show that $F \models s(q) - s(q') - 1 \geq 0$, or equivalently $F \models p - p' - 1$. Since $F \models z = 0$ and F is zero-invariant, we have $z' = 0$. Since $F \models r - r' - 1 \geq 0$ by assumption, we have $F \models (z + p) - (z' + p') - 1 \geq 0$ and so $F \models p - p' - 1 \geq 0$.

$\supseteq$ Let $r \in \langle Z \rangle + K$. Then we may write r as $z + k$ for some $z \in \langle Z \rangle$ and $k \in K$. By the definition of K, we have $k = s(q)$ for some $q \in \mathit{cone}(Y)$ such that $F \models s(q) - s(q)' - 1 \geq 0 \equiv k - k' - 1 \geq 0$. Since $q \in \mathit{cone}(Y)$ we have $k = s(q) \in \mathit{cone}(P)$ and so $F \models k \geq 0$ and thus $F \models z + k \geq 0$, so r is bounded. Since $F \models k - k' - 1 \geq 0$, $F \models z = 0$, and F is zero-stable, we have $F \models (k + z) - (k' - z') - 1$, so r is decreasing. Since r is bounded and decreasing, $r \in \mathbf{PRF}(F)$. □

Example 3. Even though Algorithm 2 is complete for synthesizing **PRF**s modulo **LIRR**, it does not find all **PRF**s with respect to the standard model. Consider

$$F \triangleq x \geq 1 \wedge y \geq 1 \wedge ((x' = 2x \wedge y' = y/2 - 1) \vee (x' = x/2 - 1 \wedge y' = 2y))$$

which admits the **PRF** xy since $F \models_{\mathbb{R}} xy \geq 1 \wedge x'y' \leq xy - 1$. However the algorithm will not find this **PRF** since we cannot derive $F \models_{\mathbf{LIRR}} xy \geq 1$ due to the fact that **LIRR** lacks axioms governing the relationship between multiplication and the order relation [18].

3.3 Proving Termination Through Polynomial Ranking Functions

This section shows how to combine the previous two subsections into a end-to-end termination analysis, which is (1) complete in the sense that it succeeds whenever the input formula has a polynomial ranking function, and (2) monotone in the sense that if $F \models G$ and the analysis finds a termination argument for G, then it can also find one for F.

Our analysis is presented in Algorithm 3, which operates by first computing a zero-stable formula and then invoking Algorithm 2 to check if it has at least one polynomial ranking function.

```
1 Function terminate-PRF(F)
      Input: An LIRR transition formula F.
      Output: Whether F admits a PRF.
2     F̂ = zero-stable-restrict(F);
3     _, _, V = prf-zero-stable(F̂);
4     if V = ∅ then
5         return unknown
6     else
7         return true
```

Algorithm 3: Proving termination through zero-stable restriction and **PRF** synthesis.

Theorem 2 (Completeness). *If F has a polynomial ranking function (modulo* **LIRR***), then Algorithm 3 returns true on F.*

Proof. Suppose F has r as a **PRF**. Since $\hat{F} = \texttt{zero-stable-restrict}(F)$ entails F (Lemma 4), r is also a **PRF** of $\hat{F}$. Letting $\langle Z, P, V\rangle = \texttt{prf-zero-inv}(\hat{F})$, we have $r \in \mathit{alg.polyhedron}(Z, P, V)$ by Theorem 1, and so V is non-empty, and Algorithm 3 returns true. □

Example 4. The reverse of Theorem 2 does not hold. Due to zero-stable restriction, Algorithm 2 can even prove termination of loops that do not admit **PRF**s even in the standard model. For example, it can prove termination of $F \triangleq x = 0 \wedge x' \neq 0$ since its zero-stable restriction is unsatisfiable. To see that F does not admit any **PRF**, suppose for a contradiction that it has r as a **PRF**. But this is impossible since there exists x' such that $r(x') > r(0) - 1$ due to the continuity of r.

The completeness of the ranking function synthesis procedures leads to several desirable properties of behavior of the resulting termination analysis, one of which is *monotonicity*, i.e., if the analysis succeeds on a transition formula G, then it is guaranteed to succeed on a stronger one F. Further, monotone termination analysis on loops can be lifted to monotone whole-program analysis by the framework presented by Zhu et al. [26].

Corollary 1 (Monotonicity). *If $F \models G$ and* `terminate-PRF` *(G) returns true, then* `terminate-PRF` *(F) returns true.*

4 Lexicographic Polynomial Ranking for Integer Transitions

In this section, we show how to synthesize lexicographic polynomial ranking functions. The strategy (inspired by [3]) is based on the connection between **WLPRF** and quasi-ranking functions. We can describe the set of quasi-ranking functions as the intersection of the sets of bounded and non-increasing polynomials of F:

$$\mathbf{QPRF}(F) = \mathit{Bounded}(F) \cap \mathit{Noninc}(F)$$

where

$$\begin{aligned} \mathit{Bounded}(F) &\triangleq \{p \in \mathbb{Q}[X] : F \models p \geq 0\} \\ \mathit{Noninc}(F) &\triangleq \{p \in \mathbb{Q}[X] : F \models p \geq p'\} \ . \end{aligned}$$

In the following, we first show how to synthesize **QPRF**s (Sect. 4.1), using which we are able to synthesize **WLPRF**s (Sect. 4.2) to prove termination. Similar to Sect. 3, we need to compute zero-stable restriction of transition formulas to make sure that the set of ranking arguments found is complete.

4.1 Synthesizing Polynomial Quasi-Ranking Functions

Algorithm 4 finds all **QPRF**s for a zero-stable transition formula F, using a variation of our strategy for finding **PRF**s.

Theorem 3 (Soundness and completeness of Algorithm 4). *Suppose F is a zero-stable transition formula. Then*

$$\mathbf{QPRF}(F) = \mathit{alg.cone}\,(\texttt{qprf-zero-stable}(F)) \ .$$

Proof. Let Z, P, Y, f, R, V, P_X be as in Algorithm 4. Let $s : \mathbb{Q}[Y] \to \mathbb{Q}[X]$ be the homomorphism mapping $y_p \mapsto p$ (corresponding to the substitution on line 6), and observe that for any linear combination of the Y variables $q = \sum_{p \in P} a_p y_p$, we have $f(q) = s(q) - s(q)'$ (as in Theorem 1). By construction (lines 5-8) we have

$$\mathit{cone}(P_X) = \{s(q) : q \in \mathit{cone}(Y), F \models f(q) \geq 0\} \ ,$$

Function `qprf-zero-stable(`F`)`
  Input: A zero-stable transition formula $F \in \mathbf{TF}(X)$.
  Output: The algebraic cone of all **QPRF**s of F.
  $\langle Z, P\rangle \leftarrow$ `consequence(`F, X`)`;
  $Y \leftarrow \{y_p : p \in P\}$ be a set of fresh variables;
  $f \leftarrow$ the ring homomorphism $\mathbb{Q}[Y] \to \mathbb{Q}[X]$ defined by $f(y_p) = p - p'$;
  /* *For any $t \in cone(R)$, we have $F \models f(t) \geq 0$.* */
  $\langle R', V'\rangle \leftarrow$ `linearize(inverse-hom(consequence(`F, $X \cup X'$`)`, f, Y`))`;
  /* *$\langle R_L, V_L\rangle$ represents the polyhedron of linear terms with positive coefficients for variables and constant coefficient 0.* */
  $R_L \leftarrow \{y : y \in Y\}, V_L \leftarrow \{0\}$;
  /* *The intersection of V-rep(R', V') and V-rep(R_L, V_L) is a cone.* */
  $\langle R, \_\rangle \leftarrow$ `intersect(`R', V', R_L, V_L`)`;
  $P_X \leftarrow \{r[y_p \mapsto p : p \in P] : r \in R\}$;
  return $\langle Z, P_X\rangle$

Algorithm 4: Computing **QPRF** for zero-stable transitions.

which by the above observation can be written equivalently as

$$cone(P_X) = \{s(q) : q \in cone(Y), F \models s(q) - s(q)' \geq 0\} \ .$$

Since $\{s(q) : q \in cone(Y)\}$ is precisely $cone(P)$, we have

$$cone(P_X) = \{p \in cone(P) : F \models p - p' \geq 0\}$$

We must show that $\mathbf{QPRF}(F) = \langle Z\rangle + cone(P_X)$. We prove inclusion in both directions.

$\subseteq$ Let $r \in \mathbf{QPRF}(F)$. Since $F \models r \geq 0$ and $alg.cone(Z, P) = \mathbf{C}_X(F)$, we must have $r = z + p$ for some $z \in \langle Z\rangle$ and $p \in cone(P)$. It is sufficient to show that $p \in cone(P_X)$. Since F is zero-stable and $F \models z = 0$, we have $F \models z - z' = 0$ and so we must have $F \models p - p' \geq 0$. It follows from the above that $p \in cone(P_X)$.

$\supseteq$ Since $\mathbf{QPRF}(F)$ is a cone it is closed under addition, so it is sufficient to prove that $\langle Z\rangle \subseteq \mathbf{QPRF}(F)$ and $cone(P_X) \subseteq \mathbf{QPRF}(F)$. Since F is zero-stable, we have $\langle Z\rangle = \{z \in \mathbb{Q}[X] : F \models z = 0\} \subseteq \mathbf{QPRF}(F)$. Since $cone(P_X) = \{p \in cone(P) : F \models p - p' \geq 0\}$, we have that each $p \in cone(P)$ is both bounded ($p \in cone(P)$) and non-increasing ($F \models p - p' \geq 0$), and thus belongs to $\mathbf{QPRF}(F)$. □

□

4.2 Lexicographic Polynomial Ranking Functions

Given Algorithm 1 for computing zero-stable restrictions and Algorithm 4 for finding **QPRF**s, we present Algorithm 5 for proving termination by finding **WLPRF**s.

```
1  Function terminate-lprf(F)
2      Z ← ∅;
3      repeat
4          Z′ ← Z;
5          ⟨Z, P⟩ ← qprf-zero-stable(zero-stable-restrict(F));
6          F ← F ∧ ⋀_{z∈Z} z′ = z ∧ ⋀_{p∈P} p′ = p;
7      until ⟨Z⟩ = ⟨Z′⟩;
8      if 1 ∈ ⟨Z⟩ then
9          return true /* F is unsatisfiable modulo LIRR iff 1 ∈ ⟨Z⟩ */
10     else
11         return unknown
```

Algorithm 5: Proving termination by synthesizing lexicographic polynomial ranking functions.

Ignoring the effects of zero-stable restriction, Algorithm 5 iteratively computes a sequence of algebraic cones that represent all **QPRF**s, and finally checks if all transitions in F have been ranked.

Example 5. Consider the transition formula

$$F : x - xy \geq 0 \wedge y \geq 0 \wedge ((x' = x \wedge y' = y - 1) \vee (y \geq 1 \wedge x' = x - 1 \wedge y' = y))$$

(which has a dimension-2 **WLPRF** $\langle y, x - xy \rangle$). The following table depicts the execution of Algorithm 5, displaying a (simplified) transition formula F, zero polynomials Z, and positive polynomials P after each iteration of the loop, culminating in $F = \mathit{false}$, which indicates that F terminates.

	F	Z	P
Before	$x - xy \geq 0 \wedge y \geq 0$ $\wedge ((x' = x \wedge y' = y - 1)$ $\vee (y \geq 1 \wedge x' = x - 1 \wedge y' = y))$	$\emptyset$	-
Iter 1	$x - xy \geq 0 \wedge y \geq 0$ $\wedge (y \geq 1 \wedge x' = x - 1 \wedge y' = y))$	$\emptyset$	$\{y\}$
Iter 2	*false*	$\emptyset$	$\{y, x - xy\}$

Theorem 4 (Correctness of Algorithm 5). *Algorithm 5 is a terminating procedure, and for any transition formula F for which* `terminate-lprf`$(F) =$ *true, we have that F terminates over the integers.*

Proof. Let $F^{(k)}, Z^{(k)}, P^{(k)}$ denote the values of F, Z, and P at the beginning of k-th iteration of the loop in Algorithm 5. We first prove termination of the algorithm. Suppose the loop does not terminate, then $\langle Z^{(k+1)} \rangle \supsetneq \langle Z^{(k)} \rangle$ for all iterations k. We have thus obtained an infinite and strictly ascending chain of ideals in the polynomial ring $\mathbb{Q}[X \cup X']$, contradicting Hilbert's basis theorem.

Now we show that if Algorithm 5 returns *true*, all integer executions of F terminate. We prove this by induction on n, the number of times the loop runs in Algorithm 5. The base case holds when $n = 1$ since the zero-stable restriction of F being unsatisfiable modulo **LIRR** implies that F terminates. Suppose that the proposition is true for $n \geq 1$ and we want to prove the case of $(n + 1)$. Consider the first iteration of the loop in Algorithm 5. For convenience, we use F to denote $F^{(1)}$, $\hat{F}$ to denote the zero-stable restriction of F, and F' to denote $F \wedge \bigwedge_{z \in Z^{(1)}} z' = z \wedge \bigwedge_{p \in P^{(1)}} p' = p$. By the inductive hypothesis, F' terminates. Suppose for a contradiction that F does not terminate. By Lemma 4 we know $\hat{F}$ also does not terminate. Define $r = \sum_{p \in P^{(1)}} p$, then $r \in \mathbf{QPRF}(\hat{F})$ due to Theorem 3. By Lemma 2, $\hat{F} \wedge r' = r$ has an infinite integer execution $x_0, x_1, \ldots$ since $\hat{F}$ has one. Let $i \in \mathbb{N}$ be arbitrary. Since $x_i \rightarrow_{\hat{F} \wedge r' = r} x_{i+1}$, we know that $\sum_{p \in P^{(1)}} p(x_{i+1}) - p(x_i) = 0$. This is a sum of nonpositive terms because $p(x_{i+1}) \leq p(x_i)$ holds for any $p \in P^{(1)}$ due to $p \in \mathbf{QPRF}(\hat{F})$. Thus for all $p \in P^{(1)}$, it holds that $p(x_{i+1}) = p(x_i)$. Since $\hat{F}$ is zero-stable, we have that $z(x_{i+1}) = z(x_i) = 0$ for all $z \in Z^{(1)}$. Thus we have $x_i \rightarrow_{F'} x_{i+1}$ and subsequently $x_0, x_1, \ldots$ is an infinite integer execution of F', contradicting the inductive hypothesis that F' terminates. □

The following theorem states that even though we operate modulo **LIRR**, we have a guarantee on the capability of the ranking functions synthesized that it is no less powerful than LLRF modulo the standard linear integer arithmetic, under mild assumptions.

Theorem 5 (Subsumption of LLRFs). *If $F \in \mathbf{TF}(X)$ is a negation-free formula involving only linear polynomials and F has an LLRF modulo linear integer arithmetic (**LIA**), then $F \wedge \bigwedge_{x \in (X \cup X')} Int(x)$ has a* **WLPRF** *modulo* **LIRR**.

Proof. We first prove a lemma as follows. Let $F(Y)$ be a ground, negation-free, **LIA** transition formula over variable set Y. Then for any affine term r over Y, if $F \models_{\mathbf{LIA}} r \geq 0$ then $F \wedge \bigwedge_{y \in Y} Int(y) \models_{\mathbf{LIRR}} r \geq 0$. (The proof is similar to Theorem 8 in [18]. Without loss of generality we assume F is a conjunctive formula. Suppose $F \models_{\mathbf{LIA}} r \geq 0$. By [11,24] there is a cutting-plane proof of $r \geq 0$ from F. Since each inference rule in a cutting-plane proof is valid **LIRR**, we have that $F \wedge \bigwedge_{y \in Y} Int(y) \models_{\mathbf{LIRR}} r \geq 0$.)

Suppose that **LIA** formula F admits an LLRF $r_1, \ldots, r_n$ of dimension n, then $F \models_{\mathbf{LIA}} r_i \geq 0$ for each i (bounded), $F \wedge \bigwedge_{j=1}^{i-1} r'_j = r_j \models_{\mathbf{LIA}} r'_i \leq r_i$ for each i (decreasing), and $F \wedge \bigwedge_{j=1}^{n} r'_j = r_j \models_{\mathbf{LIA}} \mathit{false}$ (coverage). Since the left hand side of all the implications listed above contains ground linear formulas without negation and the right hand side all contains linear inequalities (with *false* being interpreted as $0 \leq -1$), these implications also hold modulo **LIRR** by the lemma. Therefore, $r_1, \ldots, r_n$ is a **WLPRF** of F. □

Algorithm 5 is also complete w.r.t. the existence of **WLPRF**s, since it finds a **WLPRF** if there exists one for the transition formula. Moreover, it is optimal in terms of the dimension of the **WLPRF** found.

Theorem 6 (Completeness of Algorithm 5 w.r.t. WLPRF). *If a transition formula F admits a* **WLPRF** *of dimension N, then* `termination-lprf`(F) *returns true and Algorithm 5 terminates in no more than N iterations.*

Proof. Suppose that $r_1, \ldots, r_N$ is a **WLPRF** for F. Let $F^{(k)}$ denote the value of F after the kth iteration of the while loop in Algorithm 5, with the convention that if the loop exits after m iterations then $F^{(m)} = F^{(m+1)} = \cdots$. For any k, let $\langle Z^{(k)}, P^{(k)} \rangle \triangleq$ `qprf-zero-stable`(`zero-stable-restrict`$(F^{(k)})$).

We prove that $r_i \in \mathit{alg.cone}(Z^{(i)}, P^{(i)})$ for all i, by induction on i. For the base case, r_1 is a quasi ranking function for F and so also a quasi ranking function for the zero-stable restriction F, and thus $r_1 \in \mathit{alg.cone}(Z^{(1)}, P^{(1)})$. For the inductive step, we have $r_j \in \mathit{alg.cone}(Z^{(j)}, P^{(j)})$ for all $j \leq i$, and we must prove $r_{i+1} \in \mathit{alg.cone}(Z^{(i+1)}, P^{(i+1)})$. By the inductive hypothesis, we have $r_1, \ldots, r_i \in \mathit{alg.cone}(Z^{(i)}, P^{(i)})$. It follows that $F^{(i+1)} \models F^{(i)} \wedge \bigwedge_{j=1}^{i} r'_j - r_j$, so by the (Decreasing) condition of **WLPRF**, r_{i+1} is a quasi ranking function of $F^{(i)}$. It follows that r_{i+1} is a quasi ranking function of `zero-stable-restrict`$(F^{(i)})$, and thus r_{i+1} belongs to $\mathit{alg.cone}(Z^{(i+1)}, P^{(i+1)})$.

By the (Coverage) condition of **WLPRF**, we have that $F \wedge \bigwedge_{j=1}^{N} r'_j = r_j$ is unsatisfiable. Since for each j we have

$$r_j \in \mathit{alg.cone}(Z^{(j)}, P^{(j)}) \subseteq \mathit{alg.cone}(Z^{(N)}, P^{(N)})$$

we must have that $F^{(N)}$ is unsatisfiable, and so $F^{(N)} \models 1 = 0$, and thus `termination-lprf` (F) returns *true*. □

Corollary 2 (Monotonicity of Algorithm 5). *Let F and G be transition formulas with $F \models G$. If* `termination-lprf`$(G) =$ *true, then it is guaranteed that* `termination-lprf`$(F) =$ *true.*

5 Evaluation

We consider two key research questions in the experimental evaluation. First, how does the proposed technique perform in proving termination of linear or nonlinear programs comparing to existing tools, in terms of running time and the number of tasks solved. We thus compare the proposed techniques with other *sound* and *static* provers for termination. In particular, we compare against Ultimate Automizer [15,16] and 2LS [9], which are the top two *sound* tools in the `Termination` category in the 12th Competition on software verification (SV-COMP 2023). We also report a qualitative comparison with the dynamic tool DynamiTe [19]. Second, we have shown in Theorem 5 that LPRF subsumes LLRF synthesis for proving termination under certain assumptions, but we would like to understand the performance overhead of our more general procedure. We compare with the LLRF synthesis procedure implemented in ComPACT [26].

Implementation. We implement polynomial ranking functions synthesis (Sect. 3) and lexicographic polynomial ranking function synthesis (Sect. 4) as two mortal precondition operators (i.e., an operator that takes in a transition formula representing a single loop and outputs sufficient terminating conditions for that loop) in the ComPACT termination analysis framework [26], also utilizing the **LIRR** solver, consequence finding, inverse homomorphism, and nonlinear invariant generation procedures from Kincaid et al. [18]. Given any loop, we first try synthesizing polynomial ranking functions, and only attempt to synthesize lexicographic polynomial ranking function upon failure. Our implementation is denoted by "LPRF" in the tables. We have also combined our technique with phase analysis, a technique for improving termination analyzers by analyzing phase transition structures implemented in ComPACT [26].

Environment. We ran all experiments in a virtual machine with Lubuntu 22.04 LTS (kernel version 5.12) with a CPU of Intel Core i7-9750H @ 2.60 GHz and 8 GB of memory. The SV-COMP 2023 binaries of Ultimate Automizer v0.2.2-2329fc70 and 2LS version 0.9.6-svcomp23 are used in the experiments. All tools were run under a time limit of 2 min.

Benchmarks. We collected tasks from the SV-COMP 2023 `Termination` benchmarks. Since the focus of the proposed technique is to prove termination of nonlinear programs, we divide the tasks into two suites according to whether they require nonlinear reasoning. The `linear` suite consists of *terminating* and *nonrecursive* integer programs from the `Termination-MainControlFlow` subcategory in the SV-COMP, excluding the `termination-nla` folder. The `nonlinear` suite contains terminating programs without overflow[4] in the `termination-nla` folder. This suite was originally presented in [19] and contains only integer programs.

Comparing Against Sound and Static Analyses. The results of running all experiments are presented in Table 1. For the `nonlinear` suite, our proposed techniques for synthesizing polynomial ranking functions and lexicographic ranking arguments perform significantly better than the current static analysis tools in terms of both number of tasks proved and running speed. Our technique subsumes linear lexicographic ranking function synthesis for a large class of integer variable programs, and thus remains competitive for the `linear` suite. We see that there is a moderate slowdown comparing to linear lexicographic ranking function synthesis implemented in ComPACT. As a top competitor in the SV-COMP, Ultimate Automizer proves the most tasks in the `linear` suite, while requiring more time to run compared to our techniques (see the cactus plots Figs. 1 and 2).

[4] Our technique assumes unbounded integers but 2LS is bit-precise and requires this constraint.

Table 1. Experimental results on termination verification benchmarks comparing our technique (LPRF) with lexicographic linear ranking function (LLRF) synthesis, both techniques with phase analysis (+Φ), as well as ComPACT, Ultimate Automizer, and 2LS. The #c row counts the number of solved tasks, t reports total running time in seconds, excluding timeouts (# timeouts in parentheses).

#tasks		`linear`	`nonlinear`
		171	26
LPRF	#c	118	**17**
	t	333.2 (2)	**47.8** (0)
LPRF+Φ	#c	132	**17**
	t	426.0 (2)	119.5 (0)
LLRF	#c	120	3
	t	**74.6** (0)	161.1 (1)
LLRF+Φ	#c	138	4
	t	98.6 (0)	263.0 (2)
ComPACT	#c	140	4
	t	105.7 (0)	288.8 (2)
UAutomizer	#c	**160**	1
	t	2423.1 (6)	1282.7 (8)
2LS	#c	114	0
	t	5399.1 (43)	2748.7 (20)

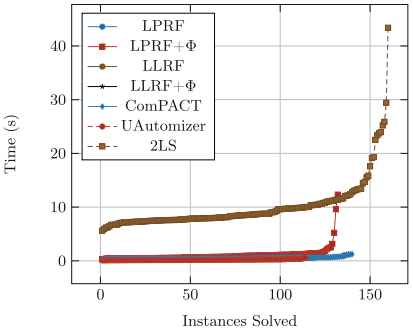

Fig. 1. Linear benchmarks.

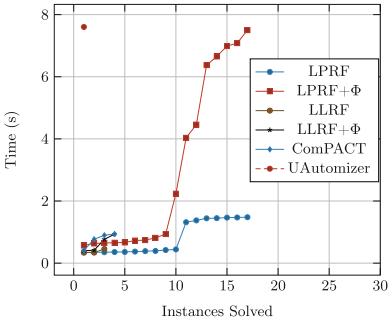

Fig. 2. Nonlinear benchmarks. 2LS cannot solve any task in the suite and is thus omitted in the plot.

Comparing Against DynamiTe. The DynamiTe paper [19] presents a *dynamic* technique that can guess and verify linear or quadratic ranking functions for nonlinear programs and proposes a benchmark suite `termination-nla` for termination of nonlinear programs. Due to hardware constraints, we could not reproduce the original evaluations for DynamiTe in our evaluation environment. Instead, we perform a comparison with the results reported in the paper. Since our tool is automated and sound but can only prove termination, we only count the terminating programs for which DynamiTe can automatically validate the discovered ranking functions. In the `termination-nla` suite, DynamiTe can learn the ranking function for most tasks (23 out of 26) but can only automatically validate 7 of them, whereas our static analysis technique LPRF is able to automatically prove 17. This observation demonstrates that verifying a given ranking function modulo nonlinear integer arithmetic is not only difficult in theory but remains challenging for modern arithmetic theory solvers. This provides additional motivation for the introduction of the weak arithmetic theory **LIRR** in this work.

6 Related Work

Ranking Function Synthesis. For linear loops, there are *complete* procedures for synthesizing particular classes of ranking functions such as linear [3,23], lexicographic linear [3,4], multi-phase [2], and nested [20]. For nonlinear loops, it is usually necessary to start with a template, e.g., polyranking functions based on a finite tree of differences between terms [5], or limiting the degree of the polynomial ranking functions to be considered [7,19]. Other procedures for synthesizing (bounded-degree) polynomial ranking functions rely on semidefinite programming [13] and cylindrical algebraic decomposition [10], but we have not found implementations for these techniques to compare with experimentally. Chatterjee et al. [8] synthesizes polynomial ranking supermartingales for probabilistic programs through Positivestellensatz, which bears some resemblance to our approach based on **LIRR** consequence finding. One key advantage of our work comparing to previous work is the completeness and monotonicity guarantee.

Decision Procedures for Termination. The decision problem for termination of linear loops was introduced by Tiwari [25]. General procedures for loops over the reals was developed by Tiwari [25], over the rationals by Braverman [6], and over the integers by Hosseini et al. [17]. Time complexity for linear and lexicographic linear ranking function synthesis has also been studied [3]. For nonlinear loops, it has been shown that termination of certain restricted classes of single-path polynomial loops over the reals are decidable, e.g., when the guard is compact and connected [21], when the loop is triangular weakly nonlinear [14], when the guard is compact semi-algebraic and the body contains continuous semi-algebraic updates [22]. Additionally, Neumann et al. [22] presents a non-constructive method for reasoning about termination via polynomial ranking functions of unbounded degree. The authors have not found any work that handles polynomial loops over *integers* without assuming real relaxations.

Acknowledgements. This work was supported in part by the NSF under grant number 1942537. Opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

1. Asadi, A., Chatterjee, K., Fu, H., Goharshady, A.K., Mahdavi, M.: Polynomial reachability witnesses via stellensätze. In: PLDI 2021, pp. 772–787. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3453483.3454076
2. Ben-Amram, A.M., Doménech, J.J., Genaim, S.: Multiphase-linear ranking functions and their relation to recurrent sets. In: Chang, B.-Y.E. (ed.) SAS 2019. LNCS, vol. 11822, pp. 459–480. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32304-2_22
3. Ben-Amram, A.M., Genaim, S.: Ranking functions for linear-constraint loops. J. ACM **61**(4), 26:1–26:55 (2014). https://doi.org/10.1145/2629488

4. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 491–504. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_48
5. Bradley, A.R., Manna, Z., Sipma, H.B.: The polyranking principle. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1349–1361. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_109
6. Braverman, M.: Termination of integer linear programs. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 372–385. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_34
7. Carbonneaux, Q., Hoffmann, J., Shao, Z.: Compositional certified resource bounds. In: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 467–478. ACM, Portland OR USA, June 2015. https://doi.org/10.1145/2737924.2737955
8. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through positivstellensatz's. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 3–22. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_1
9. Chen, H.Y., David, C., Kroening, D., Schrammel, P., Wachter, B.: Bit-precise procedure-modular termination analysis. ACM Trans. Programming Lang. Syst. **40**(1), 1–38 (2018). https://doi.org/10.1145/3121136
10. Chen, Y., Xia, B., Yang, L., Zhan, N., Zhou, C.: Discovering non-linear ranking functions by solving semi-algebraic systems. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) ICTAC 2007. LNCS, vol. 4711, pp. 34–49. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75292-9_3
11. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **306**(10), 886–904 (2006). https://doi.org/10.1016/j.disc.2006.03.009
12. Cook, B., Podelski, A., Rybalchenko, A.: Termination proofs for systems code. In: Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 415–426. ACM, Ottawa, Ontario, Canada, June 2006. https://doi.org/10.1145/1133981.1134029
13. Cousot, P.: Proving program invariance and termination by parametric abstraction, Lagrangian relaxation and semidefinite programming. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 1–24. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30579-8_1
14. Hark, M., Frohn, F., Giesl, J.: Termination of triangular polynomial loops. Formal Methods in System Design, pp. 1–63 (2023)
15. Heizmann, M., et al.: Ultimate Automizer and the CommuHash Normal Form: (Competition Contribution). In: Tools and Algorithms for the Construction and Analysis of Systems: 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22–27, 2023, Proceedings, Part II, pp. 577–581. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30820-8_39
16. Heizmann, M., Hoenicke, J., Podelski, A.: Refinement of trace abstraction. In: Palsberg, J., Su, Z. (eds.) SAS 2009. LNCS, vol. 5673, pp. 69–85. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03237-0_7
17. Hosseini, M., Ouaknine, J., Worrell, J.: Termination of Linear Loops over the Integers (Track B: Automata, Logic, Semantics, and Theory of Programming). In: DROPS-IDN/v2/Document/10.4230/LIPIcs.ICALP.2019.118. Schloss-Dagstuhl - Leibniz Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.118

18. Kincaid, Z., Koh, N., Zhu, S.: When less is more: consequence-finding in a weak theory of arithmetic. Proceedings of the ACM on Programming Languages **7**(POPL), 1275–1307 (Jan 2023). https://doi.org/10.1145/3571237
19. Le, T.C., Antonopoulos, T., Fathololumi, P., Koskinen, E., Nguyen, T.: DynamiTe: Dynamic termination and non-termination proofs. Proc. ACM Programming Lang. **4**(OOPSLA), 189:1–189:30 (2020). https://doi.org/10.1145/3428257
20. Leike, J., Heizmann, M.: Ranking templates for linear loops. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 172–186. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_12
21. Li, Y.: Termination of single-path polynomial loop programs. In: Sampaio, A., Wang, F. (eds.) ICTAC 2016. LNCS, vol. 9965, pp. 33–50. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46750-4_3
22. Neumann, E., Ouaknine, J., Worrell, J.: On Ranking Function Synthesis and Termination for Polynomial Programs. In: DROPS-IDN/v2/Document/10.4230/LIPIcs.CONCUR.2020.15. Schloss-Dagstuhl - Leibniz Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.15
23. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 239–251. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_20
24. Schrijver, A.: On Cutting Planes. In: Hammer, P.L. (ed.) Annals of Discrete Mathematics, Combinatorics 79, vol. 9, pp. 291–296. Elsevier, January 1980. https://doi.org/10.1016/S0167-5060(08)70085-2
25. Tiwari, A.: Termination of linear programs. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 70–82. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_6
26. Zhu, S., Kincaid, Z.: Termination analysis without the tears. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2021, pp. 1296–1311. Association for Computing Machinery, New York, June 2021. https://doi.org/10.1145/3453483.3454110