Logical Methods in Computer Science
Vol. 11(1:16)2015, pp. 1–27
www.lmcs-online.org

Submitted Nov. 12, 2014
Published Mar. 31, 2015

RANKING TEMPLATES FOR LINEAR LOOPS *

JAN LEIKE [a] AND MATTHIAS HEIZMANN [b]

[a] The Australian National University
e-mail address: jan.leike@anu.edu.au

[b] University of Freiburg
e-mail address: heizmann@informatik.uni-freiburg.de

Abstract. We present a new method for the constraint-based synthesis of termination arguments for linear loop programs based on *linear ranking templates*. Linear ranking templates are parameterized, well-founded relations such that an assignment to the parameters gives rise to a ranking function. Our approach generalizes existing methods and enables us to use templates for many different ranking functions with affine-linear components. We discuss templates for *multiphase*, *nested*, *piecewise*, *parallel*, and *lexicographic* ranking functions. These ranking templates can be combined to form more powerful templates. Because these ranking templates require both strict and non-strict inequalities, we use Motzkin's transposition theorem instead of Farkas' lemma to transform the generated $\exists\forall$-constraint into an $\exists$-constraint.

1. Introduction

The scope of this work is the constraint-based synthesis of termination arguments. In our setting, we consider *linear loop programs*, which are specified by a boolean combination of affine-linear inequalities over the program variables. This allows for both, deterministic and non-deterministic updates of the program variables. An example of a linear loop program is given in Figure 1.

Usually, linear lasso programs do not occur as stand-alone programs. Instead, they are used as a finite representation of an infinite path in a control flow graph. For example, in (potentially spurious) counterexamples in termination analysis [CPR06, BCF13, HLNR10, KST$^+$08, KSTW10, PR04b, PR05, HHP14], non-termination analysis [GHM$^+$08], stability analysis [CFKP11, PW07], or cost analysis [AAGP11, GZ10].

2012 ACM CCS: [**Theory of computation**]: Semantics and reasoning—Program reasoning—Program verification; [**Software and its engineering**]: Software organization and properties—Software functional properties—Formal methods—Software verification.

Key words and phrases: Linear lasso program, linear loop program, termination, linear ranking template, well-founded relation, multiphase ranking function, nested ranking function, piecewise ranking function, lexicographic ranking function, parallel ranking function, Farkas' lemma, Motzkin's theorem.

* An earlier version of this paper appeared in TACAS 2014 [LH14].

[b] This work is supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR14 AVACS).

DOI:10.2168/LMCS-11(1:16)2015

$$\begin{array}{ll} \textbf{while}\ \ (q > 0)\,: & \qquad q > 0 \\ \quad q\ :=\ q - y\,; & \qquad \wedge\, q' = q - y \\ \quad y\ :=\ y + 1\,; & \qquad \wedge\, y' = y + 1 \end{array}$$

Figure 1: A linear loop program given as program code (left) and as a formula defining a binary relation (right).

We introduce the notion of *linear ranking templates* (Section 3). These are parameterized relations specified by linear inequalities such that any assignment to the parameters yields a well-founded relation. This notion is general enough to encompass all existing methods for linear loop programs that use constraint-based synthesis of ranking functions of various kinds (see Section 8 for an assessment). Moreover, ours is the first method for synthesis of lexicographic ranking functions that does not require a mapping between loop disjuncts and lexicographic components.

In this paper we present the following linear ranking templates.

- The *multiphase ranking template* specifies a ranking function that proceeds through a fixed number of phases in the program execution. Each phase is ranked by an affine-linear function; when this function becomes non-positive, we move on to the next phase (Subsection 4.1). We call such a ranking function a multiphase ranking function.
- The *nested ranking template* specifies a ranking function that is a special case of a multiphase ranking function (Subsection 4.2). In contrast to the multiphase ranking template, the nested ranking template requires only linear constraint solving.
- The *piecewise ranking template* specifies a ranking function that is a piecewise affine-linear function with affine-linear predicates to discriminate between the pieces (Subsection 4.3).
- The *lexicographic ranking template* specifies a lexicographic ranking function that corresponds to a tuple of affine-linear functions together with a lexicographic ordering on the tuple (Subsection 4.4).
- The *parallel ranking template* targets programs that have to complete a finite number of independent tasks with no predetermined order (Subsection 4.5).

Furthermore, our linear ranking templates can be used as a 'construction kit' for composing linear ranking templates that enable more complex ranking functions (Section 5). Thus, variations on the linear ranking templates presented here can be used and completely different templates could be conceived.

Our method is described in Section 6 and can be summarized as follows. The input is a linear loop program as well as a linear ranking template. From these we construct a constraint on the parameters of the template. This constraint is a quantified nonlinear SMT formula. With Motzkin's theorem [Sch99] we transform the constraint into a purely existentially quantified constraint. This $\exists$-constraint is then passed to an SMT solver which checks its satisfiability. A positive result implies that the program terminates. Furthermore, a satisfying assignment yields a ranking function, which constitutes a termination argument for the given linear loop program.

Related approaches invoke Farkas' lemma for the transformation into $\exists$-constraints [ADFG10, BMS05a, BMS05b, CSS03, HHLP13, PR04a, Ryb10, SSM04]. Several of our ranking templates contain both strict and non-strict inequalities, yet only non-strict inequalities can be transformed using Farkas' lemma. We solve this problem by introducing

the use of Motzkin's Transposition Theorem, a generalization of Farkas' lemma. As a side effect, this also enables both strict and non-strict inequalities in the program syntax. To our knowledge, all of the aforementioned methods can be extended to programs with strict inequalities if Motzkin's theorem is applied instead of Farkas' lemma.

Our method is complete in the following sense. If there is a ranking function of the form specified by the given linear ranking template, then our method will discover this ranking function. In other words, the existence of a solution is never lost in the process of transforming the constraint.

In contrast to some related methods [HHLP13, PR04a] the constraint we generate is not linear, but rather a nonlinear algebraic constraint. Theoretically, this constraint can be decided in exponential time [GV88]. Much progress on nonlinear SMT solvers has been made and present-day implementations routinely solve nonlinear constraints of various sizes [JM12].

A related setting to linear loop programs are *linear lasso programs* (see Figure 3). These consist of a linear loop program and a program stem, both of which are specified by boolean combinations of affine-linear inequalities over the program variables. Our method can be extended to linear lasso programs through the addition of affine-linear inductive invariants, analogously to related approaches [BMS05a, CSS03, HHLP13, SSM04] (Section 7).

In this work, we consider variables with values in the rational or real numbers. Our method can be applied directly to integer programs, but in this case our completeness result does not hold. However, if we compute the integral hull of transition relations analogously to [CKRW13, HHLP13], we obtain the same completeness result for integer-valued linear loop programs as for rational- and real-valued linear loop programs.

This journal article is an extension of a conference paper [LH14]. In the conference paper, we introduced the notion of ranking templates and showed how to solve them using Motzkin's theorem (Section 4 and Section 6). We discussed the multiphase, the piecewise, and the lexicographic ranking template. The main additions in this article are the nested and parallel ranking template, the composition of ranking templates, and the extension of our method to linear lasso programs, as well as additional examples.

2. Preliminaries

In this paper we use $\mathbb{K}$ to denote a field that is either the rational numbers $\mathbb{Q}$ or the real numbers $\mathbb{R}$.

2.1. Set Theory.

We use the the following notions from set theory [Jec06]. A set X is *transitive* iff every element of X is a subset of X. A relation $R \subseteq X \times X$ is *well-founded* iff every non-empty subset of X has an R-minimal element.

Definition 2.1 (Ordinal Number [Jec06, Def. 2.10])**.** A set α is an *ordinal number* (an *ordinal*) iff it is transitive and $\in$ ('element-of') is a well-founded total order on α.

The ordinal numbers are a method of counting 'beyond infinity'. The smallest ordinal is the empty set, and for every ordinal α there is a unique successor ordinal $\alpha \cup \{\alpha\}$. The finite ordinals coincide with the natural numbers, therefore we use them interchangeably. The smallest infinite ordinal is denoted by ω. Ordinals can be added, multiplied and exponentiated, but in general these operations are not commutative.

We use the following theorem which allows us to define functions recursively.

Theorem 2.2 (Recursion Theorem [Jec06, Thm. 6.11])**.** *Let R be a well-founded relation on the set X and let G be a function on sets. Then there is a unique function F with domain X such that for every $x \in X$,*

$$F(x) = G(x, F|_{\{z \in X \mid (z,x) \in R\}}),$$

where $F|_Y$ is the restriction of the function F to the domain Y.

2.2. **Linear Loop Programs.** In this work, we consider programs that consist of a single loop. We use binary relations over the program's states to define its transition relation.

We denote by x the vector of n variables $(x_1, \dots, x_n)^T \in \mathbb{K}^n$ corresponding to program states, and by $x' = (x'_1, \dots, x'_n)^T \in \mathbb{K}^n$ the variables of the next state.

Definition 2.3 (Linear Loop Program)**.** A *linear loop program* $\textsc{loop}(x, x')$ is a binary relation defined by a formula with the free variables x and x' of the form

$$\bigvee_{i \in I} \left(A_i \binom{x}{x'} \leq b_i \ \wedge \ C_i \binom{x}{x'} < d_i \right)$$

for some finite index set I, some matrices $A_i \in \mathbb{K}^{2n \times m_i}$, $C_i \in \mathbb{K}^{2n \times k_i}$, and some vectors $b_i \in \mathbb{K}^{m_i}$ and $d_i \in \mathbb{K}^{k_i}$. The linear loop program $\textsc{loop}(x, x')$ is called *conjunctive* iff there is only one disjunct, i.e., $\#I = 1$.

Geometrically the relation $\textsc{loop}$ corresponds to a union of convex polyhedra.

Definition 2.4 (Termination)**.** A linear loop program $\textsc{loop}(x, x')$ *terminates* iff the relation $\textsc{loop}(x, x')$ is well-founded.

In general, the termination of linear loop programs is undecidable because linear loop programs can be used to simulate counter machines. An undecidability proof for the termination of linear lasso programs is given in [Lei13, Thm. 3.18].

Example 2.5. Consider the following program code.

```
while (q > 0):
    if (y > 0):
        q := q - y - 1;
    else:
        q := q + y - 1;
```

We represent this code using the following linear loop program:

$$\begin{aligned} &(q > 0 \ \wedge \ y > 0 \ \wedge \ y' = y \ \wedge \ q' = q - y - 1) \\ \vee\ &(q > 0 \ \wedge \ y \leq 0 \ \wedge \ y' = y \ \wedge \ q' = q + y - 1) \end{aligned}$$

This linear loop program is not conjunctive. Furthermore, there is no infinite sequence of states $x_0, x_1, \dots$ such that for all $i \geq 0$, the two successive states (x_i, x_{i+1}) are contained in the relation $\textsc{loop}$. Hence the relation $\textsc{loop}(x, x')$ is well-founded and the linear loop program terminates. We note that this linear loop program does not have a linear ranking function. However, termination of this program can be proven using ranking functions that we present in Subsection 4.1 and in Subsection 4.2. $\lozenge$

3. Ranking Templates

A *ranking template* is a template for a well-founded relation. More specifically, it is a parameterized formula defining a relation that is well-founded for all assignments to the parameters. If we show that a given program's transition relation LOOP is a subset of an instance of this well-founded relation, it must be well-founded itself and thus we have a proof for the program's termination. Moreover, an assignment to the parameters of the template gives rise to a ranking function. In this work, we consider ranking templates that can be encoded with linear arithmetic.

We call a formula whose free variables contain x and x' a *relation template*. Each free variable other than x and x' in a relation template is called *parameter*. Given an assignment ν to all parameter variables of a relation template $\mathrm{T}(x, x')$, the evaluation $\nu(\mathrm{T})$ is called an *instantiation of the relation template* T. We note that each instantiation of a relation template $\mathrm{T}(x, x')$ defines a binary relation.

When specifying templates, we use parameter variables to define affine-linear functions. For notational convenience, we write $f(x)$ instead of the term $s_f^T x + t_f$, where $s_f \in \mathbb{K}^n$ and $t_f \in \mathbb{K}$ are parameters. We call f an *affine-linear function symbol*.

Definition 3.1 (Linear Ranking Template)**.** Let $\mathrm{T}(x, x')$ be a relation template with parameters D and affine-linear function symbols F that can be written as a boolean combination of atoms of the form

$$\sum_{f \in F} \big(\alpha_f \cdot f(x) + \beta_f \cdot f(x')\big) + \sum_{\delta \in D} \gamma_\delta \cdot \delta \ \rhd\ 0,$$

where $\alpha_f, \beta_f, \gamma_\delta \in \mathbb{K}$ are constants and $\rhd \in \{\geq, >\}$. We call T a *linear ranking template* over D and F iff every instantiation of T defines a well-founded relation.

Example 3.2. We call the following template with parameters $D = \{\delta\}$ and affine-linear function symbols $F = \{f\}$ the *PR ranking template* [PR04a].

$$\delta > 0 \ \wedge\ f(x) > 0 \ \wedge\ f(x') < f(x) - \delta \tag{3.1}$$

In the remainder of this section, we introduce a formalism that allows us to show that every instantiation of the PR ranking template defines a well-founded relation. Let us now check the additional syntactic requirements for (3.1) to be a linear ranking template:

$$\begin{aligned} \delta > 0 \ &\equiv\ \big(0 \cdot f(x) + 0 \cdot f(x')\big) + 1 \cdot \delta > 0 \\ f(x) > 0 \ &\equiv\ \big(1 \cdot f(x) + 0 \cdot f(x')\big) + 0 \cdot \delta > 0 \\ f(x') < f(x) - \delta \ &\equiv\ \big(1 \cdot f(x) + (-1) \cdot f(x')\big) + (-1) \cdot \delta > 0 \end{aligned} \qquad \Diamond$$

The next lemma states that we can prove termination of a given linear loop program by checking that this program's transition relation is included in an instantiation of a linear ranking template.

Lemma 3.3 (Termination)**.** *Let* LOOP *be a linear loop program and let* T *be a linear ranking template with parameters D and affine-linear function symbols F. If there is an assignment ν to D and F such that the formula*

$$\forall x, x'.\ \big(\mathrm{LOOP}(x, x') \to \nu(\mathrm{T})(x, x')\big) \tag{3.2}$$

is valid, then the program LOOP *terminates.*

Proof. By definition, $\nu(\mathrm{T})$ is a well-founded relation and (3.2) is valid iff the relation LOOP is a subset of $\nu(\mathrm{T})$. Thus LOOP must be well-founded. □

In order to establish that a relation template which is conforming to the syntactic requirements is indeed a ranking template, we have to show that each instantiation of the relation template is well-founded. According to the following lemma, we can do this by showing that each assignment to D and F gives rise to a ranking function. A similar argument was given in [BA09]; we provide a significantly shortened proof by use of the Recursion Theorem, along the lines of [Jec06, Ex. 6.12].

Definition 3.4 (Ranking Function)**.** Given a binary relation R over a set Σ, a function ρ from Σ to some ordinal α is a *ranking function* for R iff for all $x, x' \in \Sigma$ the following implication holds.

$$(x, x') \in R \implies \rho(x) > \rho(x')$$

Lemma 3.5 (Existence of Ranking Functions)**.** *A binary relation R is well-founded if and only if there exists a ranking function for R.*

Proof. Let ρ be a ranking function for R. The image of a sequence decreasing with respect to R under ρ is a strictly decreasing ordinal sequence. Because the ordinals are well-ordered, this sequence cannot be infinite.

Conversely, the graph $G = (\Sigma, R)$ with vertices Σ and edges R is acyclic by assumption. Hence the function ρ that assigns to every element of Σ an ordinal number such that $\rho(x) = \sup\{\rho(x')+1 \mid (x, x') \in R\}$ is well-defined and exists due to the Recursion Theorem. □

Example 3.6. Consider the terminating linear loop program LOOP from Example 2.5. A ranking function for LOOP is $\rho : \mathbb{R}^2 \to \omega$, defined as follows.

$$\rho(q, y) = \begin{cases} \lceil q \rceil, & \text{if } q > 0, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

where $\lceil \cdot \rceil$ denotes the ceiling function that assigns to every real number r the smallest natural number that is larger or equal to r. Since we consider the natural numbers to be a subset of the ordinals, the ranking function ρ is well-defined. ◊

We use assignments to a template's parameters and affine-linear function symbols to construct a ranking function. These functions are real-valued and we transform them into ordinal-valued functions as follows.

Definition 3.7 (Ordinal Ranking Equivalent)**.** Given an affine-linear function f and a real number $\delta > 0$ called the *step size*, we define the *ordinal ranking equivalent* of f as

$$\widehat{f}(x) = \begin{cases} \left\lceil \frac{f(x)}{\delta} \right\rceil, & \text{if } f(x) > 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Our notation does not explicitly refer to δ to increase readability. In our presentation the step size δ is always clear from the context in which an ordinal ranking equivalent $\widehat{f}$ is used.

Example 3.8. Consider the linear loop program LOOP(x, x') from Example 2.5. For $\delta = 1/2$ and $f(q) = q + 1$, the ordinal ranking equivalent of f with step size δ is

$$\widehat{f}(q, y) = \begin{cases} \lceil 2(q+1) \rceil, & \text{if } q + 1 > 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

◊

The assignment from Example 3.8 to δ and f makes the implication (3.2) valid. In order to invoke Lemma 3.3 to show that the linear loop program given in Example 2.5 terminates, we need to prove that the PR ranking template is a linear ranking template. We use the following technical lemma.

Lemma 3.9 (Well-Foundedness of Ordinal Ranking Equivalents)**.** *Let f be an affine-linear function of step size $\delta > 0$ and let x and x' be two states. If $f(x) > 0$ and $f(x) - f(x') > \delta$, then $\widehat{f}(x) > 0$ and $\widehat{f}(x) > \widehat{f}(x')$.*

Proof. From $f(x) > 0$ follows that $\widehat{f}(x) > 0$. Therefore $\widehat{f}(x) > \widehat{f}(x')$ in the case $\widehat{f}(x') = 0$. For $\widehat{f}(x') > 0$, we use the fact that $f(x) - f(x') > \delta$ to conclude that $f(x)/\delta - f(x')/\delta > 1$ and hence $\widehat{f}(x') > \widehat{f}(x)$. □

Corollary 3.10. *The PR ranking template is a linear ranking template.*

Proof. Any assignment ν to δ and f satisfies the requirements of Lemma 3.9. Consequently, $\widehat{f}$ is a ranking function for $\nu(\mathrm{T})$, and by Lemma 3.5 this implies that $\nu(\mathrm{T})$ is well-founded. □

The goal of this paper is to use linear ranking templates to prove the termination of linear lasso programs, as exposed in Lemma 3.3. We defer the explanation how the $\exists\forall$-formula (3.2) can be transformed so that it is easier to solve to Section 6. The next two sections focus on additional examples for linear ranking templates.

4. Examples of Ranking Templates

4.1. **The Multiphase Ranking Template.** The multiphase ranking template targets programs that go through a finite number of phases in their execution. Each phase is ranked with an affine-linear function and the phase is considered to be completed once this function becomes non-positive.

Example 4.1. Consider the linear loop program from Figure 1 on page 2. Every execution can be partitioned into two phases: first y increases until it is positive and then q decreases until the loop condition $q > 0$ is violated. Depending on the initial values of y and q, one or more phases might be skipped altogether. ♢

Definition 4.2 (Multiphase Ranking Template)**.** We define the *k-phase ranking template* with parameters $D = \{\delta_1, \ldots, \delta_k\}$ and affine-linear function symbols $F = \{f_1, \ldots, f_k\}$ as follows.

$$\bigwedge_{i=1}^{k} \delta_i > 0 \tag{4.1}$$

$$\wedge \bigvee_{i=1}^{k} f_i(x) > 0 \tag{4.2}$$

$$\wedge\ f_1(x') < f_1(x) - \delta_1 \tag{4.3}$$

$$\wedge \bigwedge_{i=2}^{k} \Big(f_i(x') < f_i(x) - \delta_i \ \vee\ f_{i-1}(x) > 0 \Big) \tag{4.4}$$

We say that the multiphase ranking function given by an assignment to $f_1, \dots, f_k$ and $\delta_1, \dots, \delta_k$ *is in phase* i iff $f_i(x) > 0$ and $f_j(x) \leq 0$ for all $j < i$. The condition (4.2) states that there is always some i such that the multiphase ranking function is in phase i. Conditions (4.3) and (4.4) state that if we are in a phase $\geq i$, then f_i has to be decreasing by at least $\delta_i > 0$. Thus we start in phase 1 and transition through the phases $2, \dots, k$, possibly skipping some or all of them.

Two special cases of the multiphase ranking template have been discussed previously in the literature: the 1-phase ranking template, because it coincides with the PR ranking template, and the 2-phase ranking template [BM13].

Moreover, multiphase ranking functions are related to *eventually negative expressions* introduced by Bradley, Manna, and Sipma [BMS05b]. However, in contrast to our approach, they require a template tree that specifies in detail how each loop disjunct interacts with each phase.

Example 4.3. Consider the program from Figure 1 on page 2. The assignment

$$f_1(q, y) = 1 - y, \qquad f_2(q, y) = q + 1, \qquad \delta_1 = \delta_2 = \tfrac{1}{2}$$

yields a 2-phase ranking function for this program. This program is in phase 1 iff $y < 1$ and it is in phase 2 iff $y \geq 1$ and $q > -1$. ◊

Theorem 4.4. *The k-phase ranking template is a linear ranking template.*

Proof. The k-phase ranking template conforms to the linear ranking template's syntactic requirements. Let ν be an assignment to the parameters D and the affine-linear function symbols F of the k-phase ranking template $\mathrm{T}_{k\text{-phase}}$. Consider the following ranking function with codomain $\omega \cdot k$.

$$\rho(x) := \begin{cases} \omega \cdot (k - i) + \widehat{f}_i(x) & \text{if } f_j(x) \leq 0 \text{ for all } j < i \text{ and } f_i(x) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Let $(x, x') \in \nu(\mathrm{T}_{k\text{-phase}})$. By Lemma 3.5, we need to show that $\rho(x') < \rho(x)$. From (4.2) follows that $\rho(x) > 0$. Moreover, there is an i such that $f_i(x) > 0$ and $f_j(x) \leq 0$ for all $j < i$. By (4.3) and (4.4), we obtain $f_j(x') \leq 0$ for all $j < i$, because $f_j(x') < f_j(x) - \delta_j \leq 0 - \delta_j < 0$, since $f_\ell(x) \leq 0$ for all $\ell < j$.

If $f_i(x') \leq 0$, then $\rho(x') \leq \omega \cdot (k - i) < \omega \cdot (k - i) + \widehat{f}_i(x) = \rho(x)$. Otherwise, $f_i(x') > 0$ and from (4.4) follows $f_i(x') < f_i(x) - \delta_i$. By Lemma 3.9, $\widehat{f}_i(x) > \widehat{f}_i(x')$ for the ordinal ranking equivalent of f_i with step size δ_i. Hence

$$\rho(x') = \omega \cdot (k - i) + \widehat{f}_i(x') < \omega \cdot (k - i) + \widehat{f}_i(x) = \rho(x).$$

Therefore Lemma 3.5 implies that $\nu(\mathrm{T}_{k\text{-phase}})$ is well-founded. □

Does each terminating linear loop program have a multiphase ranking function if we restrict ourselves to conjunctive linear loop programs? The following theorem gives a negative answer to this question.

Theorem 4.5. *The following terminating conjunctive linear loop program does not have a multiphase ranking function.*

$$a > b \ \wedge \ b > 1 \ \wedge \ a' = 2a \ \wedge \ b' = 3b \tag{4.6}$$

Proof. The variables a and b are positive and grow exponentially, but b grows faster than a. For any input, b eventually becomes larger than a and then the loop program terminates.

Assume the loop program (4.6) has a multiphase ranking function. Then there are $\alpha_1, \beta_1, \gamma_1, \ldots, \alpha_k, \beta_k, \gamma_k \in \mathbb{R}$ such that $f_i(a,b) = \alpha_i a + \beta_i b + \gamma_i$ for all $1 \leq i \leq k$. Choose

$$b := \max\{2\} \cup \left\{ \tfrac{\gamma_i}{\beta_i}, \tfrac{-\gamma_i}{\beta_i} \mid \beta_i \neq 0 \right\}, \qquad a := \max\{b+1\} \cup \left\{ -2\beta_i b, \tfrac{-\gamma_i}{\alpha_i} \mid \alpha_i \neq 0 \right\}.$$

As maxima over a finite nonempty set, $a, b \in \mathbb{R}$ exist uniquely and we have $a > b > 1$ by construction. By setting $a' = 2a$ and $b' = 3b$, we get $(a, b, 2a, 3b) \in \textsc{loop}$. Let j be the smallest index such that $f_j(a,b) > 0$, which exists due to (4.2). According to (4.1) we obtain $\delta_j > 0$ and since j is minimal, we get $f_j(a',b') < f_j(a,b)$ from (4.4) ((4.3) in case $j = 1$). Hence

$$0 > f_j(a',b') - f_j(a,b) = 2\alpha_j a + 3\beta_j b - \alpha_j a - \beta_j b = \alpha_j a + 2\beta_j b. \tag{4.7}$$

We do an exhaustive case analysis over α_j and β_j, and show that all cases yield contradictions. Thus our assumption that there is a multiphase ranking function must be false.

(i) $\alpha_j > 0$: From (4.7) and $a \geq -2\beta_j b$ we get

$$0 > \alpha_j a + 2\beta_j b \geq -2\beta_i b + 2\beta_j b = 0.$$

(ii) $\beta_j > 0$: From (4.7) and $b \geq \gamma_j/\beta_j$ we get

$$0 > \alpha_j a + 2\beta_j b \geq \alpha_j a + \beta_j b + \beta_j \tfrac{\gamma_j}{\beta_j} = f_j(a,b) > 0.$$

(iii) $\alpha_j = \beta_j = 0$: From (4.7) we get $0 > \alpha_j a + 2\beta_j b = 0$.

(iv) $\alpha_j < 0$ and $\beta_j \leq 0$: From $a \geq -\gamma_j/\alpha_j$ we get

$$0 < f_j(a,b) = \alpha_j a + \beta_j b + \gamma_j \leq \alpha_j a + \gamma_j \leq \alpha_j \tfrac{-\gamma_j}{\alpha_j} + \gamma_j = 0.$$

(v) $\beta_j < 0$ and $\alpha_j \leq 0$: From $b \geq -\gamma_j/\beta_j$ we get

$$0 < f_j(a,b) = \alpha_j a + \beta_j b + \gamma_j \leq \beta_j b + \gamma_j \leq \beta_j \tfrac{-\gamma_j}{\beta_j} + \gamma_j = 0.$$

□

Example 4.6. Recall that we allowed our linear loop programs to have nondeterministic variable assignments. Because of this, the existence of a multiphase ranking function does not imply an upper bound on the execution time of the program. Consider the following linear loop program.

$$\begin{aligned} &(q > 0 \ \wedge \ y > 0 \ \wedge \ y' = 0) \\ \vee\ &(q > 0 \ \wedge \ y \leq 0 \ \wedge \ y' = y - 1 \ \wedge \ q' = q - 1) \end{aligned}$$

For a given input with $y > 0$, we cannot give an upper bound on the execution time: after the first loop execution, y is set to 0 and q is set to *some arbitrary value*, as no restriction to q' applies in the first disjunct. In particular, this value does not depend on the input. The remainder of the loop execution then takes $\lceil q \rceil$ iterations to terminate.

However, we can prove the program's termination with the 2-phase ranking function constructed from $f_1(q,y) = y$ and $f_2(q,y) = q$. ◊

4.2. **The Nested Ranking Template.** Like the multiphase ranking template, the nested ranking template targets programs that go through a fixed number of phases in their execution. Again, each phase has an affine-linear ranking function, but this affine-linear function cannot increase by more than the value of the previous phase's affine-linear function. Thus once the previous phase is finished, its value starts decreasing.

Definition 4.7 (Nested Ranking Template)**.** We define the *k-nested ranking template* with parameters $D = \{\delta\}$ and affine-linear function symbols $F = \{f_1, \dots, f_k\}$ as follows.

$$\delta > 0 \tag{4.8}$$
$$\wedge\ f_k(x) > 0 \tag{4.9}$$
$$\wedge\ f_1(x') < f_1(x) - \delta \tag{4.10}$$
$$\wedge \bigwedge_{i=2}^{k} f_i(x') < f_i(x) + f_{i-1}(x) \tag{4.11}$$

Example 4.8. Consider the program from Figure 1. In Example 4.3 we gave an assignment for the 2-phase ranking template. We can use almost the same assignment

$$f_1(q, y) = 1 - y, \qquad f_2(q, y) = q + 1, \qquad \delta = \tfrac{1}{2}$$

to get a 2-nested ranking function for this program. ◊

The following lemma states that nested ranking functions are a special case of multiphase ranking functions.

Lemma 4.9 (Nested Ranking Template $\subseteq$ Multiphase Ranking Template)**.** *For every assignment ν to the k-phase ranking template $\mathrm{T}_{k\text{-phase}}$ there is an assignment ν' to the k-nested ranking template $\mathrm{T}_{k\text{-nested}}$ such that $\nu'(\mathrm{T}_{k\text{-nested}}) \subseteq \nu(\mathrm{T}_{k\text{-phase}})$*

Proof. For a given ν, we choose

$$\nu'(\delta) := \nu(\delta_1), \quad \nu'(f_i) := \nu(f_i) - \nu(\delta_{i+1}) \quad \nu'(f_k) := \nu(f_k).$$

We show $\nu'(\mathrm{T}_{k\text{-nested}}) \subseteq \nu(\mathrm{T}_{k\text{-phase}})$ by showing that each of (4.8), (4.9), (4.10), and (4.11) with assignment ν' implies (4.1), (4.2), (4.3), and (4.4) with assignment ν, respectively. This is immediate for the first three lines. For (4.11) $\to$ (4.4) let (x, x') and $i > 1$ be given and assume $\nu(f_{i-1})(x) \leq 0$. We get

$$\begin{aligned}
\nu(f_i)(x') &= \nu'(f_i)(x') + \nu(\delta_{i+1}) \\
&\leq \nu'(f_i)(x') + \nu(\delta_{i+1}) - \nu(f_{i-1})(x) \\
&< \nu'(f_i)(x) + \nu'(f_{i-1})(x) + \nu(\delta_{i+1}) - \nu(f_{i-1})(x) \\
&= \nu(f_i)(x) + \nu(f_{i-1})(x) - \nu(\delta_i) - \nu(f_{i-1})(x) \\
&= \nu(f_i)(x) - \nu(\delta_i),
\end{aligned}$$

with $\nu(\delta_{k+1}) := 0$ for notational convenience. □

Theorem 4.10. *The k-nested ranking template is a linear ranking template.*

Proof. Follows from Theorem 4.4 and Lemma 4.9. □

If a nested ranking function is just a special case of a multiphase ranking function, why are we considering it separately? The advantage of the nested template is that it does not contain any disjunctions. Thus, the generated constraint can be solved using only linear constraint solving [Lei13, Ch. 6]. In our experiments, many programs that have a multiphase ranking function also have a nested ranking function. Hence it is a viable and faster alternative to the multiphase template in practice.

Example 4.11. The multiphase template is strictly more powerful than the nested template; consider the following loop program LOOP.

$$(q > 0 \ \vee \ y > 0) \ \wedge \ y' = y - 1 \ \wedge \ q' \leq q \ \wedge \ (y \leq 0 \to q' = q - 1)$$

This program has the 2-phase ranking function constructed from $f_1(q, y) = y$ and $f_2(q, y) = q$. Since there are no upper or lower bounds on q and y, only the constant function $f_i(q, y) = \gamma_i$ can be positive for all q, y. By (4.11) if f_i is constant, then f_{i-1} is positive. By induction we get that f_1 must be constant, a contradiction to (4.10). ◊

Despite their simplicity, nested ranking templates are already quite powerful. We give two nontrivial examples below that each have a nested ranking function. These ranking functions were found automatically.

Example 4.12 (Rotation53)**.** Consider the following conjunctive linear loop program.

$$q > 0 \wedge q' = q + a - 1 \wedge a' = \tfrac{3}{5}a - \tfrac{4}{5}b \wedge b' = \tfrac{4}{5}a + \tfrac{3}{5}b$$

During the execution of this loop, the vector (a, b) is rotated around 0 by the irrational angle $\arccos(3/5) \approx 53.13$ degrees. In the long run, the contribution of a to q cancels out, and q decreases on average, hence the program terminates. This program has a 3-nested ranking function constructed from the affine-linear functions

$$f_1(q, a, b) = 2q + a - 2b, \qquad f_2(q, a, b) = 4q + 5a, \quad \text{and} \quad f_3(q, a, b) = 5q. \qquad \text{◊}$$

Example 4.13 (Crazy Spirals)**.** We can also take the previous example to more extremes; consider the following conjunctive linear loop program.

$$q > 0 \wedge q' = q + a - 1 \wedge a' = 3a - 5b + c \wedge b' = 12a + 3b \wedge c' = 3c - 4d \wedge d' = 4c + 3d$$

During program execution, the vector (c, d) moves on an outward spiral centered at 0; the vector (a, b) does the same except that it is offset by c. On average, the contribution from these spirals to q cancel out, so q decreases on average. This program has a 7-nested ranking function. ◊

4.3. **The Piecewise Ranking Template.** The piecewise ranking template formalizes a ranking function that is defined piecewise using affine-linear predicates to discriminate the pieces.

Definition 4.14 (Piecewise Ranking Template)**.** We define the *k-piece ranking template* with parameters $D = \{\delta\}$ and affine-linear function symbols $F = \{f_1, \ldots, f_k, g_1, \ldots, g_k\}$ as

follows.

$$\delta > 0 \tag{4.12}$$

$$\wedge \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{k} \Big(g_i(x) < 0 \ \vee \ g_j(x') < 0 \ \vee \ f_j(x') < f_i(x) - \delta \Big) \tag{4.13}$$

$$\wedge \bigwedge_{i=1}^{k} f_i(x) > 0 \tag{4.14}$$

$$\wedge \bigvee_{i=1}^{k} g_i(x) \geq 0 \tag{4.15}$$

We call the affine-linear function symbols $\{g_i \mid 1 \leq i \leq k\}$ *discriminators* and the affine-linear function symbols $\{f_i \mid 1 \leq i \leq k\}$ *ranking pieces*.

The disjunction (4.15) states that the discriminators cover all states; in other words, the piecewise defined ranking function is a total function. Given the k different pieces $f_1, \ldots, f_k$ and a state x, we use f_i as a ranking function only if $g_i(x) \geq 0$ holds. This choice need not be unambiguous; the discriminators may overlap. If they do, we can use any one of their ranking pieces. According to (4.14), all ranking pieces are positive-valued and by (4.13), ranking piece transitions are well-defined: the rank of the new state is always less than the rank of any of the ranking pieces assigned to the old state.

Example 4.15. Consider the following linear loop program.

$$\begin{aligned} &(q > 0 \ \wedge \ p > 0 \ \wedge \ q < p \ \wedge \ q' = q - 1) \\ \vee\ &(q > 0 \ \wedge \ p > 0 \ \wedge \ p < q \ \wedge \ p' = p - 1) \end{aligned}$$

In every loop iteration, the minimum of p and q is decreased by 1 until it becomes negative. Thus, this program is ranked by the 2-piece ranking function constructed from the ranking pieces $f_1(p, q) = p$ and $f_2(p, q) = q$ with step size $\delta = 1/2$ and discriminators $g_1(p, q) = q - p$ and $g_2(p, q) = p - q$. Moreover, this program does not have a multiphase or lexicographic ranking function: both p and q may increase without bound during program execution due to non-determinism and the number of switches between p and q being the minimum value is also unbounded. ◊

Theorem 4.16. *The k-piece ranking template is a linear ranking template.*

Proof. The k-piece ranking template conforms to the linear ranking template's syntactic requirements. Let ν be an assignment to the parameter δ and the affine-linear function symbols F of the k-piece template $\mathrm{T}_{k\text{-piece}}$ be given. Consider the following ranking function with codomain ω.

$$\rho(x) := \max \big\{ \widehat{f}_i(x) \mid g_i(x) \geq 0 \big\} \tag{4.16}$$

The function ρ is well-defined, because the set $\{\widehat{f}_i(x) \mid g_i(x) \geq 0\}$ is not empty according to (4.15). Let $(x, x') \in \nu(\mathrm{T}_{k\text{-piece}})$ and let i and j be indices such that $\rho(x) = \widehat{f}_i(x)$ and $\rho(x') = \widehat{f}_j(x')$. By the definition of ρ, we have that $g_i(x) \geq 0$ and $g_j(x') \geq 0$, and (4.13) thus implies $f_j(x') < f_i(x) - \delta$. Using (4.14), we prove analogously to Lemma 3.9, that this entails $\widehat{f}_j(x') < \widehat{f}_i(x)$ and therefore $\rho(x') < \rho(x)$. Lemma 3.5 now implies that $\nu(\mathrm{T}_{k\text{-piece}})$ is well-founded. □

4.4. **The Lexicographic Ranking Template.** Lexicographic ranking functions consist of lexicographically ordered components of affine-linear functions. A state is mapped to a tuple of values such that the loop transition leads to a decrease with respect to the lexicographic ordering for this tuple. Therefore no function may increase unless a function of a lower index decreases. Additionally, at every step, there must be at least one function that decreases.

There are different definitions of lexicographic ranking functions in circulation [ADFG10, BAG13, BMS05a]; a comparison can be found in [BAG13, Sec. 2.4]. Each of these definitions for lexicographic linear ranking functions can be formalized using linear ranking templates. Here we are following the definition of [ADFG10]. This definition is the weakest, but for the other definitions the ranking template has an exponentially larger CNF, and hence our method performs comparatively poorly on them.

Definition 4.17 (Lexicographic Ranking Template)**.** We define the *k-lexicographic ranking template* with parameters $D = \{\delta_1, \dots, \delta_k\}$ and affine-linear function symbols $F = \{f_1, \dots, f_k\}$ as follows.

$$\bigwedge_{i=1}^{k} \delta_i > 0 \tag{4.17}$$

$$\wedge \bigwedge_{i=1}^{k} f_i(x) > 0 \tag{4.18}$$

$$\wedge \bigwedge_{i=1}^{k-1} \Big(f_i(x') \le f_i(x) \ \vee \bigvee_{j=1}^{i-1} f_j(x') < f_j(x) - \delta_j \Big) \tag{4.19}$$

$$\wedge \bigvee_{i=1}^{k} f_i(x') < f_i(x) - \delta_i \tag{4.20}$$

The conjunction (4.18) establishes that all lexicographic components $f_1, \dots, f_k$ have positive values. In every step, at least one component must decrease according to (4.20). From (4.19) follows that all functions corresponding to components of larger index than the decreasing function may increase.

Example 4.18. Consider the following linear loop program.

$$\begin{aligned} &(a > 0 \wedge b > 5 \wedge a' = a \wedge b' = b - 1) \\ \vee\ &(a > 0 \wedge b > 0 \wedge a' = a - 1 \wedge b' > 0) \end{aligned}$$

When taking the first disjunct, b decreases until it becomes ≤ 5. Hence we take the second disjunct eventually, decreasing a. Because a does not increase when taking the first disjunct, we can only take the second disjunct finitely many times. Since the second disjunct is always taken eventually, the program terminates.

This is proved by the 2-lexicographic ranking function constructed from the components $f_1(a, b) = a$ and $f_2(a, b) = b$. ◊

Note that the program from Example 4.18 does not have a multiphase ranking function and the program from Figure 1 on page 2 does not have a lexicographic ranking function. Thus the multiphase ranking template and the lexicographic ranking template are incomparable in expressive power.

Theorem 4.19. *The k-lexicographic ranking template is a linear ranking template.*

Proof. The k-lexicographic ranking template conforms to the linear ranking template's syntactic requirements. Let ν be an assignment to the parameters D and the affine-linear function symbols F of the k-lexicographic template $\mathrm{T}_{k\text{-lex}}$. Consider the following ranking function with codomain ω^k.

$$\rho(x) := \sum_{i=1}^{k} \omega^{k-i} \cdot \widehat{f}_i(x) \tag{4.21}$$

Let $(x, x') \in \nu(\mathrm{T}_{k\text{-lex}})$. From (4.18) follows $f_j(x) > 0$ for all j, so $\rho(x) > 0$. By (4.20) and Lemma 3.9, there is a minimal i such that $\widehat{f}_i(x') < \widehat{f}_i(x)$. According to (4.19), we have $\widehat{f}_1(x') \leq \widehat{f}_1(x)$ and hence inductively $\widehat{f}_j(x') \leq \widehat{f}_j(x)$ for all $j < i$, since i was minimal.

$$\begin{aligned}\rho(x') &= \sum_{j=1}^{k} \omega^{k-j} \cdot \widehat{f}_j(x') \leq \sum_{j=1}^{i-1} \omega^{k-j} \cdot \widehat{f}_j(x) + \sum_{j=i}^{k} \omega^{k-j} \cdot \widehat{f}_j(x') \\ &< \sum_{j=1}^{i-1} \omega^{k-j} \cdot \widehat{f}_j(x) + \omega^{k-i} \cdot \widehat{f}_i(x) \leq \rho(x)\end{aligned}$$

Therefore Lemma 3.5 implies that $\nu(\mathrm{T}_{k\text{-lex}})$ is well-founded. □

4.5. **The Parallel Ranking Template.** The parallel ranking template targets programs that do multiple tasks in parallel where progress on each task can be nondeterministic. These tasks have no predetermined order of execution. We assume that each task can be ranked by an affine-linear ranking function.

Definition 4.20 (Parallel Ranking Template)**.** We define the *k-parallel ranking template* with parameters $D = \{\delta_1, \ldots, \delta_k\}$ and affine-linear function symbols $F = \{f_1, \ldots, f_k\}$ as follows.

$$\bigwedge_{i=1}^{k} \delta_i > 0 \tag{4.22}$$

$$\wedge \bigwedge_{i=1}^{k} f_i(x') \leq f_i(x) \tag{4.23}$$

$$\wedge \bigvee_{i=1}^{k} \Big(f_i(x) > 0 \ \wedge \ f_i(x') < f_i(x) - \delta_i \Big) \tag{4.24}$$

The ranking functions $f_1, \ldots, f_k$ correspond to k different tasks. The conjunction (4.23) states that none of the ranking functions may increase at any point. Moreover, (4.24) states that with every transition, at least one task has to make progress and end in a finite number of steps. Note that (4.24) is not given in conjunctive normal form (CNF). When transformed in CNF, the number of conjuncts blows up exponentially.

Example 4.21. Consider the following linear loop program.

$$\begin{aligned}&(a > 0 \ \wedge \ a' = a - 1 \ \wedge \ b' = b) \\ \vee\ &(b > 0 \ \wedge \ b' = b - 1 \ \wedge \ a' = a)\end{aligned}$$

This programs performs two tasks nondeterministically in parallel: the first task takes a iterations and the second task takes b iterations; the program terminates once both tasks have been completed. The 2-parallel ranking function constructed from $f_1(a,b) = a$ and $f_2(a,b) = b$ proves this program terminating. ◊

Theorem 4.22. *The k-parallel ranking template is a linear ranking template.*

Proof. The k-parallel ranking template conforms to the linear ranking template's syntactic requirements. Let ν be an assignment to the parameters D and the affine-linear function symbols F of the k-parallel template $\mathrm{T}_{k\text{-parallel}}$. Consider the following ranking function with codomain ω.

$$\rho(x) := \sum_{i=1}^{k} \widehat{f_i}(x) \tag{4.25}$$

Let $(x, x') \in \nu(\mathrm{T}_{k\text{-parallel}})$. From (4.23) follows $f_i(x') \leq f_i(x)$ for all i, so $\widehat{f_i}(x') \leq \widehat{f_i}(x)$. By (4.24), there is a j such that $f_j(x) > 0$ and $f_j(x') < f_j(x) - \delta_j$. Therefore we have $\widehat{f_j}(x') < \widehat{f_j}(x)$ by Lemma 3.9. Hence

$$\rho(x') = \sum_{i=1}^{k} \widehat{f_i}(x') = \widehat{f_j}(x') + \sum_{i\neq j} \widehat{f_i}(x') \leq \widehat{f_j}(x') + \sum_{i\neq j} \widehat{f_i}(x) < \widehat{f_j}(x) + \sum_{i\neq j} \widehat{f_i}(x) = \rho(x).$$

Now Lemma 3.5 implies that $\nu(\mathrm{T}_{k\text{-parallel}})$ is well-founded. □

5. Composition of Templates

In this section we discuss how more powerful linear ranking templates can be constructed based on the linear ranking templates from Section 4. First, we consider a program that is terminating, but whose termination cannot be proven using one of the ranking templates presented so far.

Example 5.1. Consider the following linear loop program.

$$\begin{aligned}&(q > 0 \wedge y > 0 \wedge y' = y - 1 \wedge q' = q \wedge x' = x)\\ \vee\ &(q > 0 \wedge y \leq 0 \wedge q' = q - x \wedge x' = x + 1)\end{aligned}$$

When executing the first disjunct, y decreases until it becomes negative. Then we execute the second disjunct: we increment x, set y to some arbitrary value, and decrement q if x is positive. If y was reset to some positive value, the first disjunct is executed again, but the values of q and x do not change until the second disjunct is executed. Eventually, x is positive and from then on q is decremented until it is nonpositive; thus the program terminates. ◊

The program's behavior resembles a lexicographic ranking function with q as the first component and y as the second. However, q is decremented only after x becomes positive: the first component needs 2 phases. We want a ranking template for a lexicographic ranking function that has multiphase ranking functions instead of affine-linear functions in every component. How can we construct a linear ranking template for such a ranking function?

Observe that all of our linear ranking templates share the following subformulas.

(i) $f(x') \leq f(x)$
(ii) $f(x') < f(x) - \delta$

(iii) $f(x) > 0$

In the context of ranking templates, these formulas have the following meaning.

(i) The function f is non-increasing.
(ii) The function f is decreasing.
(iii) The codomain of the function f is well-founded.

Here f is always an affine-linear function. The idea of template composition is to replace the subformulas (i–iii) with subformulas of the same meaning for more powerful functions. We next define triples of formulas that are suitable substituents for (i–iii), called *composed template recipes*. Afterwards, we will use composed template recipes to build new linear ranking templates.

Definition 5.2 (Composed Template Recipe)**.** A *composed template recipe* is defined recursively according to the following rules.

(C1) The *PR template recipe* $(\mathrm{T}^{\leq}_{PR}, \mathrm{T}^{<}_{PR}, \mathrm{T}^{>0}_{PR})$ is a composed template recipe with

$$\begin{aligned}\mathrm{T}^{\leq}_{\mathrm{PR}} &\equiv f(x') \leq f(x)\\ \mathrm{T}^{<}_{\mathrm{PR}} &\equiv f(x') < f(x) - \delta \ \wedge \ \delta > 0\\ \mathrm{T}^{>0}_{\mathrm{PR}} &\equiv f(x) > 0.\end{aligned}$$

(C2) The *k-piece template recipe* $(\mathrm{T}^{\leq}_{k\text{-piece}}, \mathrm{T}^{<}_{k\text{-piece}}, \mathrm{T}^{>0}_{k\text{-piece}})$ is a composed template recipe with

$$\mathrm{T}^{\leq}_{k\text{-piece}} \equiv \bigvee_{i=1}^{k} g_i(x) \geq 0 \ \wedge \ \bigwedge_{i=1}^{k}\bigwedge_{j=1}^{k} \Big(g_i(x) < 0 \ \vee \ g_j(x') < 0 \ \vee \ f_j(x') \leq f_i(x)\Big)$$

$$\mathrm{T}^{<}_{k\text{-piece}} \equiv \delta > 0 \ \wedge \ \bigvee_{i=1}^{k} g_i(x) \geq 0 \ \wedge \ \bigwedge_{i=1}^{k}\bigwedge_{j=1}^{k} \Big(g_i(x) < 0 \vee g_j(x') < 0 \vee f_j(x') < f_i(x) - \delta\Big)$$

$$\mathrm{T}^{>0}_{k\text{-piece}} \equiv \bigwedge_{i=1}^{k} f_i(x) > 0.$$

(C3) Given k composed template recipes $(\mathrm{T}^{\leq}_1, \mathrm{T}^{<}_1, \mathrm{T}^{>0}_1), \ldots, (\mathrm{T}^{\leq}_k, \mathrm{T}^{<}_k, \mathrm{T}^{>0}_k)$ which do not share any parameters or affine-linear function symbols, we can construct a composed template recipe $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$ according to one of the following three composition rules.

Composition rule	$\mathrm{T}^{\leq}$	$\mathrm{T}^{<}$	$\mathrm{T}^{>0}$
k-phase	$\mathrm{T}^{\leq}_1 \wedge \bigwedge_{i>1}(\mathrm{T}^{\leq}_i \vee \mathrm{T}^{>0}_{i-1})$	$\mathrm{T}^{<}_1 \wedge \bigwedge_{i>1}(\mathrm{T}^{<}_i \vee \mathrm{T}^{>0}_{i-1})$	$\bigvee_i \mathrm{T}^{>0}_i$
k-lexicographic	$\bigwedge_{i=1}^{k}\left(\mathrm{T}^{\leq}_i \vee \bigvee_{j=1}^{i-1} \mathrm{T}^{<}_j\right)$	$\bigvee_i \mathrm{T}^{<}_i \wedge \bigwedge_{i=1}^{k-1}\left(\mathrm{T}^{\leq}_i \vee \bigvee_{j=1}^{i-1} \mathrm{T}^{<}_j\right)$	$\bigwedge_i \mathrm{T}^{>0}_i$
k-parallel	$\bigwedge_i \mathrm{T}^{\leq}_i$	$\bigwedge_i \mathrm{T}^{\leq}_i \wedge \bigvee_i(\mathrm{T}^{<}_i \wedge \mathrm{T}^{>0}_i)$	$\bigvee_i \mathrm{T}^{>0}_i$

The intuition behind Definition 5.2 is that we build composed templates recursively using PR template recipes (C1) or k-piece template recipes (C2) as the base case and plugging them into composition rules given in (C3).

There is a composition rule for each linear ranking template presented in Section 4 but the k-piece ranking template and the k-nested ranking template. We cannot define a k-piece or a k-nested composition rule analogously, because not all of these ranking templates'

atoms are of the form (i–iii) above: they also have atoms containing multiple affine-linear function symbols ($f_i(x') < f_i(x) + f_{i-1}(x)$ in (4.11) and $f_j(x') < f_i(x) - \delta$ in (4.13)).

Given a composed template recipe $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$, we call the conjunction $\mathrm{T}^{<} \wedge \mathrm{T}^{>0}$ a *composed template.* The following theorem states that composed templates are linear ranking templates.

Theorem 5.3. *If $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$ is a composed template recipe, then the composed template $\mathrm{T}^{<} \wedge \mathrm{T}^{>0}$ is a linear ranking template.*

The proof of Theorem 5.3 is deferred to the end of this section.

Example 5.4 (The k-Phase Composition Rule)**.** We apply the k-phase composition rule to k PR template recipes. Let $D := \{\delta_i \mid 1 \leq i \leq k\}$ be parameters and let $F = \{f_i \mid 1 \leq i \leq k\}$ be affine-linear function symbols. For each i, we have three formulas $\mathrm{T}^{\leq}_{\mathrm{PR},i}$, $\mathrm{T}^{<}_{\mathrm{PR},i}$, and $\mathrm{T}^{>0}_{\mathrm{PR},i}$. Using the k-phase composition rule from Definition 5.2 (C3), we get the composed template recipe $(\mathrm{T}^{\leq}_{k\text{-phase}}, \mathrm{T}^{<}_{k\text{-phase}}, \mathrm{T}^{>0}_{k\text{-phase}})$ where

$$\begin{aligned}
\mathrm{T}^{\leq}_{k\text{-phase}} &\equiv f_1(x') \leq f_1(x) \ \wedge \bigwedge_{i=2}^{k} (f_i(x') \leq f_i(x) \ \vee \ f_{i-1}(x) > 0), \\
\mathrm{T}^{<}_{k\text{-phase}} &\equiv f_1(x') < f_1(x) - \delta_1 \ \wedge \ \delta_1 > 0 \\
&\qquad \wedge \bigwedge_{i=2}^{k} ((f_i(x') < f_i(x) - \delta_i \ \wedge \ \delta_i > 0) \ \vee \ f_{i-1}(x) > 0), \\
\mathrm{T}^{>0}_{k\text{-phase}} &\equiv \bigvee_{i=1}^{k} f_i(x) > 0.
\end{aligned}$$

By Theorem 5.3, $\mathrm{T}^{<}_{k-\mathrm{phase}} \wedge \mathrm{T}^{>0}_{k-\mathrm{phase}}$ is a linear ranking template. In fact, we already know this from Theorem 4.4, because the formula $\mathrm{T}^{<}_{k\text{-phase}} \wedge \mathrm{T}^{>0}_{k\text{-phase}}$ is equivalent to the k-phase ranking template. ◊

Remark 5.5.

(i) Let $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$ be the k-phase composition rule applied to k PR template recipes. Then the composed template $\mathrm{T}^{>0} \wedge \mathrm{T}^{<}$ is equivalent to the k-phase ranking template.

(ii) Let $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$ be the k-lexicographic composition rule applied to k PR template recipes. Then the composed template $\mathrm{T}^{>0} \wedge \mathrm{T}^{<}$ is equivalent to the k-lexicographic ranking template.

(iii) Let $(\mathrm{T}^{\leq}, \mathrm{T}^{<}, \mathrm{T}^{>0})$ be the k-parallel composition rule applied to k PR template recipes. Then the composed template $\mathrm{T}^{>0} \wedge \mathrm{T}^{<}$ is equivalent to the k-parallel ranking template.

Proof. From Definition 4.2, Definition 4.17, and Definition 4.20. □

Next, we construct a composed template to prove termination of Example 5.1.

Example 5.6. We apply the ℓ-lexicographic composition rule to ℓ copies of the composed template recipe from Example 5.4. Let $D := \{\delta_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq \ell\}$ be parameters and let $F = \{f_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq \ell\}$ be affine-linear function symbols. For each j, we apply the k-phase composition rule to k PR template recipes as in Example 5.4, using the parameters $D_j := \{\delta_{i,j} \mid 1 \leq i \leq k\}$ and the affine-linear function

symbols $F_j := \{f_{i,j} \mid 1 \leq i \leq k\}$. Let the resulting composed template recipe be denoted $(\mathrm{T}^{\leq}_{k\text{-phase},j}, \mathrm{T}^{<}_{k\text{-phase},j}, \mathrm{T}^{>0}_{k\text{-phase},j})$. Next, we apply the ℓ-lexicographic composition rule to the ℓ composed template recipes $(\mathrm{T}^{\leq}_{k\text{-phase},j}, \mathrm{T}^{<}_{k\text{-phase},j}, \mathrm{T}^{>0}_{k\text{-phase},j})$ resulting in the composed template recipe $(\mathrm{T}^{\leq}_{\mathrm{lm}}, \mathrm{T}^{<}_{\mathrm{lm}}, \mathrm{T}^{>0}_{\mathrm{lm}})$ where

$$\begin{aligned}
\mathrm{T}^{\leq}_{\mathrm{lm}} \equiv \bigwedge_{j=1}^{\ell} \Bigg(& \Big(f_{1,j}(x') \leq f_{1,j}(x) \wedge \bigwedge_{i=2}^{k} (f_{i,j}(x') \leq f_{i,j}(x) \vee f_{i-1,j}(x) > 0)\Big) \\
& \vee \bigvee_{t=1}^{j-1} \Big(f_{1,t}(x') < f_{1,t}(x) - \delta_{1,t} \wedge \delta_{1,t} > 0 \\
& \qquad \wedge \bigwedge_{i=2}^{k} ((f_{i,t}(x') < f_{i,t}(x) - \delta_{i,t} \wedge \delta_{i,t} > 0) \vee f_{i-1,t}(x) > 0)\Big)\Bigg), \\
\mathrm{T}^{<}_{\mathrm{lm}} \equiv \bigvee_{j=1}^{\ell} \Bigg(& f_{1,j}(x') < f_{1,j}(x) - \delta_{1,j} \wedge \delta_{1,j} > 0 \\
& \wedge \bigwedge_{i=2}^{k} ((f_{i,j}(x') < f_{i,j}(x) - \delta_{i,j} \wedge \delta_{i,j} > 0) \vee f_{i-1,j}(x) > 0)\Bigg) \\
& \wedge \bigwedge_{j=1}^{\ell-1} \Bigg(\Big(f_{1,j}(x') \leq f_{1,j}(x) \wedge \bigwedge_{i=2}^{k} (f_{i,j}(x') \leq f_{i,j}(x) \vee f_{i-1,j}(x) > 0)\Big) \\
& \qquad \vee \bigvee_{t=1}^{j-1} \Big(f_{1,t}(x') < f_{1,t}(x) - \delta_{1,t} \wedge \delta_{1,t} > 0 \\
& \qquad\qquad \wedge \bigwedge_{i=2}^{k} ((f_{i,t}(x') < f_{i,t}(x) - \delta_{i,t} \wedge \delta_{i,t} > 0) \vee f_{i-1,t}(x) > 0)\Big)\Bigg), \\
\mathrm{T}^{>0}_{\mathrm{lm}} \equiv \bigwedge_{j=1}^{\ell} \bigvee_{i=1}^{k} & f_{i,j}(x) > 0.
\end{aligned}$$

By Theorem 5.3, $\mathrm{T}^{<}_{\mathrm{lm}} \wedge \mathrm{T}^{>0}_{\mathrm{lm}}$ is a linear ranking template. ◊

Example 5.7. Using the composed template recipe from Example 5.6, we can find a ranking function for Example 5.1:

$$\begin{aligned}
f_{1,1}(q,x,y) &= 1 - x & f_{1,2}(q,x,y) &= q \\
f_{2,1}(q,x,y) &= y & f_{2,2}(q,x,y) &= y
\end{aligned}$$ ◊

Proof of Theorem 5.3. First, we need to check the syntactic requirements. This was already shown for the PR ranking template and the k-piece ranking template. Any substitution is a boolean combination of parts of simpler templates, and the syntactic requirements for linear ranking templates allow for arbitrary boolean combinations of atoms.

To show well-foundedness, we prove the following statement by induction over the recursive construction of the composed template recipes. We show that for all assignments ν to the parameters and affine-linear function symbols, we find a function $\rho : \Sigma \to \alpha$ from the program states Σ to some ordinal α such that

(i) $\nu(\mathrm{T}^{\leq})(x,x')$ implies $\rho(x') \leq \rho(x)$.
(ii) $\nu(\mathrm{T}^{<})(x,x')$ and $\rho(x) > 0$ imply $\rho(x') < \rho(x)$.
(iii) $\nu(\mathrm{T}^{>0})(x,x')$ implies $\rho(x) > 0$.

For the base case, we have a PR template recipe or a k-piece template recipe, and we get a ranking function $\rho : \Sigma \to \omega$. Claims (ii) and (iii) follow from Lemma 3.9 and Theorem 4.16. For the PR template recipe, claim (i) holds because $f(x') \leq f(x)$ implies $\widehat{f}(x') \leq \widehat{f}(x)$. For the k-piece template recipe, we prove this analogously to the proof of Theorem 4.16, using $f_i(x') \leq f_i(x)$ instead of $f_i(x') < f_i(x) - \delta$.

For the induction step, assume that claims (i–iii) hold for the composed template recipes $(\mathrm{T}_1^{\leq}, \mathrm{T}_1^{<}, \mathrm{T}_1^{>0})$, …, $(\mathrm{T}_k^{\leq}, \mathrm{T}_k^{<}, \mathrm{T}_k^{>0})$. Thus for every $i = 1 \dots k$, we have a ranking function $\rho_i : \Sigma \to \alpha_i$ with ordinal α_i as codomain. We consider the three inductive cases in turn.

- *k-phase*: We define the ranking function

$$\rho(x) := \begin{cases} \sum_{j=1}^{i-1} \alpha_j + \rho_i(x) & \text{if } \rho_j(x) = 0 \text{ for all } j < i \text{ and } \rho_i(x) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

  Let $(x,x') \in \nu(\mathrm{T}^{\leq})$, and let i be the current phase, i.e., $\rho_i(x) > 0$ and $\rho_j(x) = 0$ for all $j < i$. For all $j < i$, we have $(x,x') \in \nu(\mathrm{T}_j^{\leq})$ and $(x,x') \notin \nu(\mathrm{T}_j^{>0})$ by inductive hypothesis, and hence $\rho_j(x') \leq \rho_j(x) = 0$. Therefore we obtain $(x,x') \in \nu(\mathrm{T}_i^{\leq})$ and hence $\rho_i(x') \leq \rho_i(x)$ (note the subscript i instead of j), which implies $\rho(x') \leq \rho(x)$ in case $\rho_i(x') > 0$. Otherwise we have a phase transition and thus $\rho(x') = 0 < \rho(x)$ or $i < k$. In the latter case, we know $\rho_{i+1}(x') < \alpha_{i+1}$, and hence $\rho(x) > \sum_{j=1}^{i-1} \alpha_j > \sum_{j=1}^{i-2} \alpha_j + \rho_{i-2}(x') = \rho(x')$.

  For $(x,x') \in \nu(\mathrm{T}^{<})$, we analogously get $\rho_i(x') < \rho_i(x)$ and hence $\rho(x') < \rho(x)$. For $(x,x') \in \nu(\mathrm{T}^{>0})$, we have that $\rho_i(x) > 0$ for some i and hence $\rho(x) > 0$ by the induction hypothesis.
- *k-lexicographic*: We define the ranking function

$$\rho(x) := \sum_{i=1}^{k} \rho_i(x) \prod_{j=i+1}^{k} \alpha_j.$$

  Let $(x,x') \in \nu(\mathrm{T}^{\leq})$. If $(x,x') \in \nu(\mathrm{T}_i^{\leq})$ for all i, then we get $\rho_i(x') \leq \rho_i(x)$ by the induction hypothesis, and hence $\rho(x') \leq \rho(x)$. Otherwise there is an $n \leq k$ such that $(x,x') \in \nu(\mathrm{T}_n^{<})$ and $(x,x') \in \nu(\mathrm{T}_j^{\leq})$ for all $j < n$. By the induction hypothesis, we obtain $\rho_n(x') < \rho_n(x)$ and $\rho_j(x') \leq \rho_j(x)$ for all $j < n$. Since $\rho_i(x') < \alpha_i$, we have $\prod_{j=n+1}^{k} \alpha_j > \sum_{i=n+1}^{k} \rho_i(x') \prod_{j=i+1}^{k} \alpha_j$ and thus

$$\begin{aligned} \rho(x) &= \sum_{i=1}^{k} \rho_i(x) \prod_{j=i+1}^{k} \alpha_j \\ &\geq \left(\sum_{i=1}^{n-1} \rho_i(x) \prod_{j=i+1}^{k} \alpha_j \right) + \left(\rho_n(x) \prod_{j=n+1}^{k} \alpha_j \right) \\ &= \left(\sum_{i=1}^{n-1} \rho_i(x) \prod_{j=i+1}^{k} \alpha_j \right) + \left((\rho_n(x) - 1) \prod_{j=n+1}^{k} \alpha_j \right) + \prod_{j=n+1}^{k} \alpha_j \end{aligned}$$

$$> \left(\sum_{i=1}^{n-1} \rho_i(x) \prod_{j=i+1}^{k} \alpha_j\right) + \left((\rho_n(x) - 1) \prod_{j=n+1}^{k} \alpha_j\right) + \left(\sum_{i=n+1}^{k} \rho_i(x') \prod_{j=i+1}^{k} \alpha_j\right)$$
$$\geq \left(\sum_{i=1}^{n-1} \rho_i(x') \prod_{j=i+1}^{k} \alpha_j\right) + \left(\rho_n(x') \prod_{j=n+1}^{k} \alpha_j\right) + \left(\sum_{i=n+1}^{k} \rho_i(x') \prod_{j=i+1}^{k} \alpha_j\right)$$
$$= \rho(x').$$

For $(x, x') \in \nu(\mathrm{T}^{<})$, we proceed analogously except that the case $(x, x') \in \nu(\mathrm{T}_i^{\leq})$ for all i cannot occur. For $(x, x') \in \nu(\mathrm{T}^{>0})$, we get $\rho_i(x) > 0$ by the induction hypothesis, thus $\rho(x) > 0$.

- *k-parallel*: We define the ranking function

$$\rho(x) := \sum_{i=1}^{k} \rho_i(x).$$

For $(x, x') \in \nu(\mathrm{T}^{\leq})$, we have that $\rho_i(x') \leq \rho_i(x)$ for all i by the induction hypothesis, and hence $\rho(x') \leq \rho(x)$. For $(x, x') \in \nu(\mathrm{T}^{<})$, we again have by the induction hypothesis that $\rho_i(x') \leq \rho_i(x)$ for all i, and that there is an i such that $\rho_i(x') < \rho_i(x)$. Therefore we obtain $\rho(x') < \rho(x)$. For $(x, x') \in \nu(\mathrm{T}^{>0})$, we have that there is an i such that $\rho_i(x) > 0$ by the induction hypothesis, therefore $\rho(x) > 0$.

This completes the induction. To finish the proof, we note that according to claim (ii) and (iii), ρ is a ranking function for $\nu(\mathrm{T}^{<} \wedge \mathrm{T}^{>0})$, and by Lemma 3.5 this implies that $\nu(\mathrm{T}^{<} \wedge \mathrm{T}^{>0})$ is well-founded. □

Although the procedure introduced in this section allows for an infinite number of different ranking templates, it is not exhaustive. We expect that there are many more types of ranking functions that can be formalized using linear ranking templates, and possibly also composed with other templates.

6. Synthesizing Ranking Functions

Following related approaches [ADFG10, BMS05a, BMS05b, CSS03, HHLP13, PR04a, Ryb10, SSM04], we transform the ∃∀-constraint (3.2) into an ∃-constraint. This transformation makes the constraint more easily solvable not only because we remove universal quantification, but also because it reduces the number of nonlinear operations in the constraint. Every application of an affine-linear function symbol f corresponds to a nonlinear term $s_f^T x + t_f$ where s_f is a vector of real-valued parameters and t_f is a real-valued parameter. For this step, we need the following theorem.

6.1. Motzkin's Transposition Theorem.

Intuitively, Motzkin's transposition theorem states that a given system of linear inequalities has no solution if and only if a contradiction can be derived via a positive linear combination of the inequalities.

Theorem 6.1 (Motzkin's Transposition Theorem [Sch99, Cor. 7.1k])**.** *For $A \in \mathbb{K}^{m\times n}$, $C \in \mathbb{K}^{\ell\times n}$, $b \in \mathbb{K}^m$, and $d \in \mathbb{K}^\ell$, the formulas* (M1) *and* (M2) *are equivalent.*

$$\forall x \in \mathbb{K}^n.\ \neg(Ax \leq b\ \wedge\ Cx < d) \tag{M1}$$

$$\begin{aligned}\exists \lambda \in \mathbb{K}^m\ \exists \mu \in \mathbb{K}^\ell.\ \lambda \geq 0\ \wedge\ \mu \geq 0 \\ \wedge\ \lambda^T A + \mu^T C = 0\ \wedge\ \lambda^T b + \mu^T d \leq 0 \\ \wedge\ (\lambda^T b < 0\ \vee\ \mu \neq 0)\end{aligned} \tag{M2}$$

If ℓ is set to 1 in Theorem 6.1, we obtain the affine version of Farkas' lemma [Sch99, Cor. 7.1h]. Therefore Motzkin's theorem is strictly superior to Farkas' lemma, as it allows for a combination of both strict and non-strict inequalities. Moreover, it is logically optimal in the sense that it enables the transformation of *any* purely universally quantified (Π_1^0) formula from the theory of linear arithmetic.

6.2. **Constraint Transformation.** We fix a linear loop program LOOP and a linear ranking template T with parameters D and affine-linear function symbols F. For simplicity of presentation, we assume the loop program LOOP does not contain any strict inequalities, and the ranking template T does not contain any non-strict inequalities; however, recall that we are using Motzkin's theorem instead of Farkas' lemma precisely to lift this restriction. For the fully general constraints, see [Lei13, Ch. 5]. We write LOOP in disjunctive normal form and T in conjunctive normal form:

$$\begin{aligned}\textsc{loop}(x, x') &\equiv \bigvee_{i\in I} A_i \tbinom{x}{x'} \leq b_i \\ \textsc{t}(x, x') &\equiv \bigwedge_{j\in J} \bigvee_{\ell\in L_j} \textsc{t}_{j,\ell}(x, x') \equiv \bigwedge_{j\in J} \bigvee_{\ell\in L_j} t_{j,\ell}^T \tbinom{x}{x'} > e_{j,\ell}\end{aligned}$$

We prove the termination of LOOP by solving the constraint (3.2). This constraint is implicitly existentially quantified over the parameters D and the parameters corresponding to the affine-linear function symbols F.

$$\forall x, x'.\ \left(\Big(\bigvee_{i\in I} A_i \tbinom{x}{x'} \leq b_i\Big) \to \Big(\bigwedge_{j\in J} \bigvee_{\ell\in L_j} \textsc{t}_{j,\ell}(x, x')\Big)\right) \tag{6.1}$$

First, we transform the constraint (6.1) into an equivalent constraint of the form required by Motzkin's theorem.

$$\bigwedge_{i\in I} \bigwedge_{j\in J} \forall x, x'.\ \neg \left(A_i \tbinom{x}{x'} \leq b_i\ \wedge\ \Big(\bigwedge_{\ell\in L_j} \neg \textsc{t}_{j,\ell}(x, x')\Big)\right) \tag{6.2}$$

Now, Motzkin's Transposition theorem transforms the constraint (6.2) into an equivalent existentially quantified constraint:

$$\bigwedge_{i\in I} \bigwedge_{j\in J} \exists \lambda \geq 0\ \exists \zeta \geq 0.\ \lambda^T A_i + \sum_{\ell\in L_j} \zeta_\ell t_{j,\ell}^T = 0\ \wedge\ \lambda^T b_i + \sum_{\ell\in L_j} \zeta_\ell e_{j,\ell} < 0 \tag{6.3}$$

For every inequality in (M1), a new existentially quantified variable is added in (M2). These new existentially quantified variables are called *Motzkin coefficients*.

Input: linear loop program LOOP and a list of linear ranking templates $\mathcal{T}$
Output: a ranking function for LOOP or `null` if none is found

```
foreach T ∈ 𝒯 do:
    let φ = ∀x, x'. (LOOP(x, x') → T(x, x'))
    let ψ = transformWithMotzkin(φ)
    if SMTsolver.checkSAT(ψ):
        let (D, F) = T.getParameters()
        let ν = getAssignment(ψ, D, F)
        return T.extractRankingFunction(ν)
return null
```

Figure 2: Our ranking function synthesis algorithm described in pseudocode. The function `transformWithMotzkin` transforms the $\exists\forall$-constraint φ into an $\exists$-constraint ψ as described in Subsection 6.2. The ranking function is extracted from an assignment of the template with the function `extractRankingFunction`. This function returns a description of the ranking function depending on the template; for example the ordinal-based representation from the proofs.

The $\exists$-constraint (6.3) is then checked for satisfiability. If an assignment is found, it gives rise to a ranking function. Conversely, if no assignment exists, then there cannot be an instantiation of the linear ranking template and thus no ranking function of the kind formalized by the linear ranking template exists. In this sense our method is sound and complete.

Theorem 6.2 (Soundness)**.** *If the transformed $\exists$-constraint* (6.3) *is satisfiable, then the linear loop program terminates.* □

Theorem 6.3 (Completeness)**.** *If the $\exists\forall$-constraint* (3.2) *is satisfiable, then so is the transformed $\exists$-constraint* (6.3)*.* □

6.3. **Ranking Template Pools.** Our method for ranking function synthesis can be applied as follows. We fix a finite pool of linear ranking templates $\mathcal{T}$, consisting of multiphase, nested, piecewise, lexicographic, and parallel ranking templates as well as composed templates in various sizes. The input is a linear loop program LOOP that we want to check for termination. We start by picking a linear ranking template T from the pool $\mathcal{T}$. From the ranking template T we build the constraint (3.2) to the parameters and affine-linear function symbols of T. This constraint is transformed using Motzkin's theorem to an $\exists$-constraint (6.3). If this constraint is satisfiable, this gives rise to a ranking function according to Lemma 3.5, and thus we proved that the loop program LOOP terminates. Otherwise, we try again using the next linear ranking template from the pool $\mathcal{T}$ until the pool has been exhausted. If the pool has been exhausted, the proof of the loop program LOOP's termination failed. However, due to the completeness of our method, we know that the loop program LOOP does not have a ranking function of the form specified by any of the linear ranking templates in the pool. Figure 2 is a description of our method in pseudocode.

7. Linear Lasso Programs

Our method extends to the more general setting of *linear lasso programs*. These are linear loop programs that have a program stem in addition to the loop (see Figure 3). We use affine-linear inductive invariants to extract the information that is crucial for the termination proof from the stem. This is in line with related approaches [CSS03, SSM04, BMS05a, HHLP13].

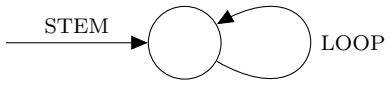

Figure 3: A lasso program.

Definition 7.1 (Linear Lasso Program)**.** A *linear lasso program* $\mathbf{P} = (\text{STEM}, \text{LOOP})$ consists of

- a linear loop program LOOP, and
- a predicate STEM, defined by a formula with the free variables x of the form

$$\bigvee_{i \in I} \left(A_i x \leq b_i \ \wedge \ C_i x < d_i \right)$$

  for some finite index set I, some matrices $A_i \in \mathbb{K}^{n \times m_i}$, $C_i \in \mathbb{K}^{n \times k_i}$, and some vectors $b_i \in \mathbb{K}^{m_i}$ and $d_i \in \mathbb{K}^{k_i}$.

The linear lasso program $\mathbf{P}$ is called *conjunctive* iff there is only one disjunct in both transitions STEM and LOOP.

Definition 7.2 (Affine-Linear Supporting Invariant)**.** A formula ψ is an *affine-linear supporting invariant* for the linear lasso program $\mathbf{P}$ iff there is an affine-linear function f such that

$$\psi(x) \equiv f(x) \rhd 0$$

with $\rhd \in \{\geq, >\}$, and the following two formulas hold.

$$\forall x.\ \text{STEM}(x) \to \psi(x) \tag{II}$$

$$\forall x, x'.\ \psi(x) \wedge \text{LOOP}(x, x') \to \psi(x') \tag{IC}$$

The affine-linear supporting invariant ψ is *strict* iff $\rhd$ is $>$ and *non-strict* otherwise.

Given a linear lasso program, we do the same transformation steps as in Subsection 6.2, adding a finite number of supporting invariants:

$$\forall x, x'.\ \text{LOOP}(x, x') \ \wedge \ \bigwedge_{\ell} \psi_\ell(x) \to \text{T}(x, x')$$

In fact, every conjunct in (6.2) gets m supporting invariants:

$$\bigwedge_{i \in I} \bigwedge_{j \in J} \forall x, x'. \neg \left(A_i \tbinom{x}{x'} \leq b_i \ \wedge \ \Big(\bigwedge_{\ell=1}^{m} \psi_{i,j,\ell}(x) \Big) \ \wedge \ \Big(\bigwedge_{\ell \in L_j} \neg \text{T}_{j,\ell}(x, x') \Big) \right) \tag{7.1}$$

To insure that the $\psi_{i,j,\ell}(x)$ are indeed supporting invariants, we add the constraints (II) and (IC) for each $(i, j, \ell) \in I \times J \times \{1, \ldots, m\}$. Each of these constraints is then transformed using Motzkin's theorem. analogously to Subsection 6.2. Not all invariants are inductive and we only consider invariants that are affine-linear inequalities. We do not retain completeness of our method in the sense of Theorem 6.3.

The invariant initiating (II) is a linear constraint, but the invariant consecution (IC) is nonlinear. We could make (IC) linear by restricting ourselves to non-decreasing invariants [HHLP13]. However, the overall constraints are generally still nonlinear because the constraints that come from the linear ranking template are generally nonlinear.

8. Related Work

Synthesis of linear ranking functions for linear loop programs was first discussed by Colón and Sipma [CS01]. This was extended to a complete template-based method by Podelski and Rybalchenko [PR04a, Ryb10], using the PR ranking template as discussed in Example 3.2. Their method is not complete over the integers. Cook et al. [CKRW13] compute the integral hull of transition relations in order obtain the same completeness for integers and bitvectors. Bagnara and Mesnard generalize the PR ranking template to the 2-phase ranking template, relying on nonlinear constraint solving [BM13].

Bradley, Manna, and Sipma propose a constraint-based approach for linear lasso programs [BMS05a]. Their termination argument is a lexicographic ranking function with each lexicographic component corresponding to one loop disjunct. This requires nonlinear constraint solving and an ordering on the loop disjuncts. The authors extend this approach in [BMS05b] by the use of *template trees*. These trees allow each lexicographic component to have a ranking function that decreases not necessarily in every step, but *eventually*.

Ben-Amram and Genaim discuss the synthesis of affine-linear and lexicographic ranking functions for linear loop programs over the integers [BAG13]. They prove that this problem is generally co-NP-complete and show that several special cases admit a polynomial time complexity.

In [CFM12] the authors also address the problem of finding termination arguments for (not necessarily conjunctive) linear loop programs. In contrast to our work, the authors do not synthesize the termination argument directly. Instead, they iteratively synthesize linear ranking functions and obtain a disjunctively well-founded relation [PR04b] as a termination argument.

Approaches for computing lexicographic linear ranking functions for a more general class of programs, namely programs that can consist of several (potentially nested) loops are presented in [ADFG10] and [CSZ13]. On linear loop programs, both algorithms involve choosing an ordering on the loop disjuncts. Hence, both approaches are either incomplete or have to use backtracking to iteratively consider all possible orderings of loop disjuncts.

Our method is not able to prove termination for all terminating linear loop programs. Termination is decidable for the subclass of deterministic conjunctive linear loop programs of the form

$$\texttt{while}(B_s x > b_s \wedge B_w x \geq b_w)\ \ x\ \texttt{:=}\ Ax + c\texttt{;}$$

where the matrices B_s, B_w, A and vectors b_s, b_w, c are rational, and variables can take on rational or real values [Tiw04]. This class also admits decidable termination analysis over the integers for the homogeneous case where $b_s, b_w, c = 0$ [Bra06]. However, their method is not targeted at the synthesis of ranking functions.

Ranking functions can also be computed via abstract interpretation [CC12]. Urban and Miné [Urb13, UM14a, UM14b] introduced the domain of piecewise defined ordinal-valued functions for this approach. In contrast to our work, their approach is applicable to programs with arbitrary structure and not restricted to linear lasso programs. However, the

	PR	k-phase	k-nested	k-piece	k-lexicographic	k-parallel
Parameters	1	k	1	1	k	k
Function symbols	1	k	k	$2k$	k	k
Conjuncts	3	$2k+1$	$k+2$	k^2+k+2	$3k$	2^k+2k
Atoms	3	$4k-1$	$k+2$	$3k^2+2k+1$	$(5k^2+k)/2$	$k2^k+2k$

Table 1: Statistics of our linear ranking templates in CNF; the integer k specifies their size. Every affine-linear function symbol constributes $n+1$ parameters to the template, where n is the number of program variables.

authors do not provide completeness results that state that a ranking function of a certain form can always be found.

9. Conclusion

We presented a sound and complete method for constraint-based synthesis of ranking functions for linear loop programs. For this method, we introduced the notion of *linear ranking templates*, which are parameterized formulas for well-founded relations. In Section 3 we established how they can be applied to prove termination (Lemma 3.3) and that an instantiation of a linear ranking template gives rise to a ranking function (Lemma 3.5). Our method can be applied to different kinds of ranking functions that previously have been considered independently (affine-linear and lexicographic ranking functions), in addition to enabling new kinds (multiphase, piecewise, and parallel ranking functions). The ranking templates can also be composed into more powerful templates, allowing for more general ranking functions. See Table 1 for statistics on the size of our ranking templates.

Our method can be applied to linear loop programs and linear lasso programs with variables that are rational numbers, real numbers, or integers. In general, it requires solving nonlinear algebraic constraints, but some linear ranking templates such as the PR ranking template or the nested ranking template only require linear constraint solving.

Acknowledgements. We wish to thank Samir Genaim for pointing our an error in the conference version of Theorem 4.5, and Amir M. Ben-Amram for detailed comments on the Master's thesis [Lei13] from which this paper was derived. Moreover, we thank Andreas Podelski for his detailed feedback and helpful suggestions.

References

[AAGP11] Elvira Albert, Puri Arenas, Samir Genaim, and Germán Puebla. Closed-form upper bounds in static cost analysis. *J. Autom. Reasoning*, 46(2):161–203, 2011.

[ADFG10] Christophe Alias, Alain Darte, Paul Feautrier, and Laure Gonnord. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *SAS*, volume 6337, pages 117–133. Springer, 2010.

[BA09] Amir M. Ben-Amram. Size-change termination, monotonicity constraints and ranking functions. In *CAV*, page 109–123. Springer, 2009.

[BAG13] Amir M. Ben-Amram and Samir Genaim. Ranking functions for linear-constraint loops. In *POPL*, 2013.

[BCF13] Marc Brockschmidt, Byron Cook, and Carsten Fuhs. Better termination proving through cooperation. In *CAV*, pages 413–429. Springer, 2013.

[BM13] Roberto Bagnara and Fred Mesnard. Eventual linear ranking functions. In *Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming*, pages 229–238. ACM, 2013.

[BMS05a] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Linear ranking with reachability. In *CAV*, pages 491–504. Springer, 2005.

[BMS05b] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. The polyranking principle. In *ICALP*, pages 1349–1361. Springer, 2005.

[Bra06] Mark Braverman. Termination of integer linear programs. In *CAV*, pages 372–385. Springer, 2006.

[CC12] Patrick Cousot and Radhia Cousot. An abstract interpretation framework for termination. In *POPL*, pages 245–258. ACM, 2012.

[CFKP11] Byron Cook, Jasmin Fisher, Elzbieta Krepska, and Nir Piterman. Proving stabilization of biological systems. In *VMCAI*, pages 134–149, 2011.

[CFM12] Hong Yi Chen, Shaked Flur, and Supratik Mukhopadhyay. Termination proofs for linear simple loops. In *SAS*, pages 422–438. Springer, 2012.

[CKRW13] Byron Cook, Daniel Kroening, Philipp Rümmer, and Christoph M. Wintersteiger. Ranking function synthesis for bit-vector relations. *Formal methods in system design*, 43(1):93–120, 2013.

[CPR06] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Terminator: Beyond safety. In *CAV*, pages 415–418, 2006.

[CS01] Michael A. Colón and Henny B. Sipma. Synthesis of linear ranking functions. In *TACAS*, pages 67–81. Springer, 2001.

[CSS03] Michael A. Colón, Sriram Sankaranarayanan, and Henny B. Sipma. Linear invariant generation using non-linear constraint solving. In *CAV*, pages 420–432. Springer, 2003.

[CSZ13] Byron Cook, Abigail See, and Florian Zuleger. Ramsey vs. lexicographic termination proving. In *TACAS*, pages 47–61. Springer, 2013.

[GHM+08] Ashutosh Gupta, Thomas A. Henzinger, Rupak Majumdar, Andrey Rybalchenko, and Ru-Gang Xu. Proving non-termination. In *POPL*, pages 147–158, 2008.

[GV88] Dmitrii Grigor'ev and Nicolai Vorobjov. Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5(1–2):37–64, 1988.

[GZ10] Sumit Gulwani and Florian Zuleger. The reachability-bound problem. In *PLDI*, pages 292–304, 2010.

[HHLP13] Matthias Heizmann, Jochen Hoenicke, Jan Leike, and Andreas Podelski. Linear ranking for linear lasso programs. In *ATVA*, 2013.

[HHP14] Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Termination analysis by learning terminating programs. In *CAV*, pages 797–813, 2014.

[HLNR10] William R. Harris, Akash Lal, Aditya V. Nori, and Sriram K. Rajamani. Alternation for termination. In *SAS*, pages 304–319, 2010.

[Jec06] Thomas Jech. *Set Theory*. Springer, 3rd edition, 2006.

[JM12] Dejan Jovanović and Leonardo De Moura. Solving non-linear arithmetic. In *IJCAR*, pages 339–354. Springer, 2012.

[KST+08] Daniel Kroening, Natasha Sharygina, Stefano Tonetta, Aliaksei Tsitovich, and Christoph M. Wintersteiger. Loop summarization using abstract transformers. In *ATVA*, pages 111–125, 2008.

[KSTW10] Daniel Kroening, Natasha Sharygina, Aliaksei Tsitovich, and Christoph M. Wintersteiger. Termination analysis with compositional transition invariants. In *CAV*, pages 89–103, 2010.

[Lei13] Jan Leike. Ranking function synthesis for linear lasso programs. Master's thesis, University of Freiburg, Germany, 2013.

[LH14] Jan Leike and Matthias Heizmann. Ranking templates for linear loops. In *TACAS*, pages 172–186. Springer, 2014.

[PR04a] Andreas Podelski and Andrey Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, pages 239–251. Springer, 2004.

[PR04b] Andreas Podelski and Andrey Rybalchenko. Transition invariants. In *LICS*, pages 32–41, 2004.

[PR05] Andreas Podelski and Andrey Rybalchenko. Transition predicate abstraction and fair termination. In *POPL*, pages 132–144, 2005.

[PW07] Andreas Podelski and Silke Wagner. A sound and complete proof rule for region stability of hybrid systems. In *HSCC*, pages 750–753, 2007.

[Ryb10] Andrey Rybalchenko. Constraint solving for program verification theory and practice by example. In *CAV*, pages 57–71. Springer, 2010.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

[SSM04] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Constraint-based linear-relations analysis. In *SAS*, pages 53–68. Springer, 2004.

[Tiw04] Ashish Tiwari. Termination of linear programs. In *CAV*, pages 70–82. Springer, 2004.

[UM14a] Caterina Urban and Antoine Miné. An abstract domain to infer ordinal-valued ranking functions. In *ESOP*, pages 412–431. Springer, 2014.

[UM14b] Caterina Urban and Antoine Miné. A decision tree abstract domain for proving conditional termination. In *SAS*, pages 302–318, 2014.

[Urb13] Caterina Urban. The abstract domain of segmented ranking functions. In *SAS*, pages 43–62. Springer, 2013.