Linear Ranking for Linear Lasso Programs ⋆

Matthias Heizmann, Jochen Hoenicke, Jan Leike, and Andreas Podelski

University of Freiburg, Germany

Abstract. The general setting of this work is the constraint-based synthesis of termination arguments. We consider a restricted class of programs called lasso programs. The termination argument for a lasso program is a pair of a ranking function and an invariant. We present the—to the best of our knowledge—first method to synthesize termination arguments for lasso programs that uses linear arithmetic. We prove a completeness theorem. The completeness theorem establishes that, even though we use only linear (as opposed to non-linear) constraint solving, we are able to compute termination arguments in several interesting cases. The key to our method lies in a constraint transformation that replaces a disjunction by a sum.

1 Introduction

Termination is arguably the single most interesting correctness property of a program. Research on proving termination can be divided according to three (interrelated) topics, namely: practical tools [1,9,13,17,18,19,21,22], decidability questions [4,8,25], and constraint-based synthesis of termination arguments [2,3,5,6,7,10,12,14,20,23]. The work in this paper falls under the research on the third topic. The general goal of this research is to investigate how one can derive a constraint from the program text and compute a termination argument (of a restricted form) through the solution of the constraint, i.e., via constraint solving.

In this paper, we present a method for the synthesis of termination arguments for a specific class of programs that we call *lasso programs*. As the name indicates, the control flow graph of a lasso program is of a restricted shape: a *stem* followed by a *loop*.

Lasso programs do not appear as stand-alone programs. Lasso programs appear in practice whenever one needs a finite representation of an infinite path in a control flow graph, for example in (potentially spurious) counterexamples in a termination analysis[13,17,18,19], non-termination analysis[16], stability analysis[11,22], or cost analysis[1,15].

Importantly, the termination argument for a lasso program is a pair of a ranking function and an invariant (the rank must decrease only for states that satisfy the invariant). Figure 1 shows an example of a lasso program.

⋆ 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)

```
y := 23;
while( x >= 0 ) {
    x := x - y;
    y := y + 1;
}
```

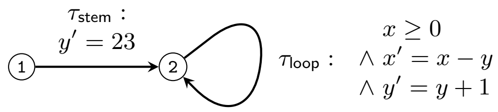


Fig. 1. Example of a lasso program and its formal representation $P_{\mathsf{yPositive}} = (\tau_{\mathsf{stem}}, \tau_{\mathsf{loop}})$. The ranking function defined by $f(x, y) = x$ decreases in transitions from states that satisfy the invariant $y \geq 1$ (the ranking function does not decrease when $y \leq 0$).

The class of lasso programs lies between two classes of programs for which constraint-based methods have been studied extensively. For the first, more specialized class, methods can be based on linear arithmetic constraint solving [2,3,10,12,20]. For the second, more general class, all known methods are based on non-linear arithmetic constraint solving [5,7]. The contribution of our method can be phrased, alternatively, as the generalization of the applicability of the 'linear methods', or as the optimization of the 'non-linear method' to a 'linear method' for a subproblem. The step from 'non-linear' to 'linear' is interesting for principled reasons (non-linear arithmetic constraint solving is undecidable in the case of integers). As we will show the step is also practically interesting.

The reader may wonder how practical tools presently handle the situation where one needs to compute termination arguments for lasso programs. One possibility is to resort to heuristics. For example, instead of computing a termination argument for the lasso program in Figure 1, one would compute the ranking function $f(x) = x$ for the program `while(x>=0){x:=x-23;}`.

The key to our method is a constraint transformation that replaces a disjunction by a sum. We apply the 'or-to-plus' transformation in the context of Farkas' Lemma. Following [2,5,10,12,20], we apply Farkas' Lemma in order to eliminate the universal quantifiers in the arithmetic constraint whose solution is the termination argument. If we apply Farkas' Lemma to the constraint *after* the 'or-to-plus' transformation, we obtain a *linear* arithmetic constraint.

The effect of the 'or-to-plus' transformation to the constraint is a restriction of its solution space. The restriction seems strong; i.e., in some cases, the solution space becomes empty. We can characterize those cases. In other words, we can characterize when the 'or-to-plus' transformation leads to the loss of an termination argument, and when it does not. The characterization is formulated as a completeness theorem for which we will present the proof. This characterization allows us to establish that, even though we use only linear (as opposed to non-linear) constraint solving, we are able to compute termination arguments in several interesting cases. A possible explanation for this (perhaps initially surprising) fact is that, for synthesis, we are interested in the mere existence of a solution, and the loss of *many* solutions does not necessarily mean the loss of *all* solutions of the constraint.

We have implemented our method and we have used our implementation to illustrate the applicability and the efficiency of our method. Our implementation is available through a web interface, together with a number of example programs (including the ones used in this paper).[1]

2 Preliminaries: Linear Arithmetic

We use $\boldsymbol{x}$ to denote the vector with entries $x_1, \dots, x_n$, and $\boldsymbol{x}^\mathsf{T}$ to denote the transposed vector of $\boldsymbol{x}$. As usual, the expression $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ denotes the conjunction of linear constraints $\bigwedge_{j=0}^{m} (\sum_{i=0}^{n} a_{ij} \cdot x_i) \leq b_j$.

We call a relation $\tau(\boldsymbol{x}, \boldsymbol{x}')$ a *linear relation* if τ is defined by a conjunction of linear constraints over the variables $\boldsymbol{x}$ and $\boldsymbol{x}'$, i.e., if there is a matrix A with m rows and $2n$ columns and a vector $\boldsymbol{b}$ of size m such that the following equation holds.

$$\tau(\boldsymbol{x}, \boldsymbol{x}') = \{(\boldsymbol{x}, \boldsymbol{x}') \mid A \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}\}$$

We call a function $f(\boldsymbol{x})$ an *(affine) linear function*, if $f(\boldsymbol{x})$ is defined by an affine linear term, i.e., there is a vector $\boldsymbol{r}^\mathsf{T}$ and a number r_0 such that the following equation holds.

$$f(\boldsymbol{x}) = \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x} + r_0.$$

We call a predicate $I(\boldsymbol{x})$ a *linear predicate*, if $I(\boldsymbol{x})$ is defined by a linear inequality, i.e., there is a vector $\boldsymbol{s}^\mathsf{T}$ and a number s_0 such that following equivalence holds.

$$I(\boldsymbol{x}) = \{\boldsymbol{x} \mid \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0\}.$$

Farkas' Lemma. We use the affine version of Farkas' Lemma [24] which is also used in [2,5,12,23,20] and states the following. Given

- a satisfiable conjunction of linear constraints $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$
- and a linear constraint $\boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \leq \delta$,

the following equivalence holds.

$$\forall \boldsymbol{x} \;\; (A \cdot \boldsymbol{x} \leq \boldsymbol{b} \rightarrow \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \leq \delta) \quad \text{iff} \quad \exists \boldsymbol{\lambda} \;\; (\boldsymbol{\lambda} \geq 0 \wedge \boldsymbol{\lambda}^\mathsf{T} \cdot A = \boldsymbol{c}^\mathsf{T} \wedge \boldsymbol{\lambda}^\mathsf{T} \cdot \boldsymbol{b} \leq \delta)$$

3 Lasso Program

To abstract away from program syntax, we define a lasso program directly by the two relations that generate its execution sequences.

[1] http://ultimate.informatik.uni-freiburg.de/LassoRanker

Definition 1 (Lasso Program). *Given a set of states Σ, a* lasso program

$$P = (\tau_{\mathsf{stem}}, \tau_{\mathsf{loop}})$$

is given by the two relations $\tau_{\mathsf{stem}} \subseteq \Sigma \times \Sigma$ and $\tau_{\mathsf{loop}} \subseteq \Sigma \times \Sigma$. We call τ_{stem} the stem *of P and τ_{loop} the* loop *of P.*

An execution of the lasso program *P is a possibly infinite sequence of states $\sigma_0, \sigma_1, \dots$ such that*

- *the pair of the first two states is an element of the stem, i.e.,*

$$(\sigma_0, \sigma_1) \in \tau_{stem}$$

- *and each other consecutive pair of states is an element of the loop, i.e.,*

$$(\sigma_i, \sigma_{i+1}) \in \tau_{loop} \qquad \textit{for } i = 1, 2, \dots$$

We call the lasso program P terminating *if P has no infinite execution.*

We use constraints over primed and unprimed variables to denote a transition relation (see Figure 1).

In order to avoid cumbersome technicalities, we consider only lasso programs that have an execution that contains at least three states. This means we consider only programs where the relational composition of τ_{stem} and τ_{loop} is non-empty, i.e.,

$$\tau_{\mathsf{stem}} \circ \tau_{\mathsf{loop}} \neq \emptyset.$$

Since Turing, a termination argument is based on an ordering which does not allow infinite decreasing chains (such as ordering on the natural numbers). Here, we use the ordering over the set of positive reals which is defined by some value $\delta > 0$, namely

$$a \prec_\delta b \quad \text{iff} \quad a \geq 0 \ \wedge \ a - b \geq \delta \qquad\qquad a, b \in \mathbb{R}.$$

Ranking Function. We call a function f from the states of the lasso program P into the reals $\mathbb{R}$ a *ranking function* for P if there is a positive number $\delta > 0$ such that for each consecutive pair of states $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ of a loop transition ($i \geq 1$) in every execution of P

- the value of f is decreasing by at least δ, i.e.,

$$f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i+1}) \geq \delta,$$

- and the value of f is non-negative, i.e.,

$$f(\boldsymbol{x}_i) \geq 0.$$

If there is a ranking function for the lasso program P, then P is terminating.

Inductive Invariant. We call a state predicate $I(\boldsymbol{x})$ an *inductive invariant* of the lasso program P if

– the predicate holds after executing the stem, i.e.,

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{stem}}(\boldsymbol{x}, \boldsymbol{x}') \to I(\boldsymbol{x}'), \tag{$\varphi_{\mathsf{invStem}}$}$$

– and if the predicate holds before executing the loop, then the predicate holds afterwards, i.e.,

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad I(\boldsymbol{x}) \ \wedge\ \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \to I(\boldsymbol{x}'). \tag{$\varphi_{\mathsf{invLoop}}$}$$

Ranking Function with Supporting Invariant. We call a pair of a ranking function $f(\boldsymbol{x})$ and an inductive invariant $I(\boldsymbol{x})$ of the lasso program P a *ranking function with supporting invariant* if the following holds.

– There exists a positive real number $\delta > 0$ such that, if the inductive invariant holds then an execution of the loop decreases the value of the ranking function by at least δ, i.e.,

$$\exists \delta > 0 \forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad I(\boldsymbol{x}) \ \wedge\ \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \to f(\boldsymbol{x}) - f(\boldsymbol{x}') \geq \delta. \tag{$\varphi_{\mathsf{rkDecr}}$}$$

– In states in which the inductive invariant holds and the loop can be executed, the value of the ranking function is non-negative, i.e.,

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad I(\boldsymbol{x}) \ \wedge\ \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \to f(\boldsymbol{x}) \geq 0. \tag{$\varphi_{\mathsf{rkBound}}$}$$

For example, the lasso program depicted in Figure 1 has the ranking function $f(x, y) = x$ with supporting invariant $y \geq 1$.

Linear lasso programs. Linear lasso programs. For the remainder of this paper we consider only linear lasso programs, linear ranking functions, and linear inductive invariants which we will define next. The variables of the programs will range over the reals until we come to Section 9 where we turn to programs over integers.

Definition 2 (Linear Lasso Program). *A* linear lasso program

$$P = (\tau_{\mathsf{stem}}, \tau_{\mathsf{loop}})$$

is a lasso program whose states are vectors over the reals, i.e. $\Sigma = \mathbb{R}^n$*, and whose relations* τ_{stem} *and* τ_{loop} *are linear relations.*

We use the expression $A_{\mathsf{stem}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{stem}}$ to denote the relation τ_{stem} of P. We use the expression $A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}}$ to denote the relation τ_{loop} of P.

Linear Ranking Function. If a ranking function $f : \mathbb{R}^n \to \mathbb{R}$ is an (affine) linear function, we call f a *linear ranking function.* We use $r_1, \ldots, r_n$ as coefficients of a linear ranking function, $\boldsymbol{r}$ as their vector,

$$f : \mathbb{R}^n \to \mathbb{R} \qquad f(\boldsymbol{x}) = \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x} + r_0.$$

Linear Invariant. If an inductive invariant $I(\boldsymbol{x})$ is a linear predicate, we call I a *linear inductive invariant.* We use $s_1, \ldots, s_n$ as coefficients of the term that defines the linear predicate, $\boldsymbol{s}$ as their vector,

$$I(\boldsymbol{x}) \;\equiv\; \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0.$$

4 The Or-to-Plus Method

Our constraint-based method for the synthesis of linear ranking functions for linear lasso programs consists of three main steps:

Step 1. Set up four (universally quantified) constraints whose free variables are the coefficients of a linear ranking function with linear supporting invariant.
Step 2. Apply Farkas' Lemma to the four constraints to obtain equivalent constraints without universal quantification.
Step 3. Obtain solutions for the free variables by linear constraint solving.

The particularity of our four constraints in Step 1 is that the application of Farkas' Lemma in Step 2 yields constraints that are linear.

Instead of presenting our constraints immediately, we derive them in three successive transformations of constraints. We start with the four constraints $(\varphi_{\mathsf{invStem}})$, $(\varphi_{\mathsf{invLoop}})$, $(\varphi_{\mathsf{rkDecr}})$, and $(\varphi_{\mathsf{rkBound}})$. Below, we have rephrased the four constraints for the setting where the ranking function is linear and the supporting invariant is linear. We marked them (φ_1^{BMS}), (φ_2^{BMS}), (φ_3^{BMS}), and (φ_4^{BMS}) in reference to Bradley, Manna and Sipma [5] who were the first to use them in the corresponding step of their method.

The Bradley–Manna–Sipma constraints
for the special case of lasso programs and one linear supporting invariant[2]

$$\begin{array}{rrr}
\forall \boldsymbol{x}\, \forall \boldsymbol{x}' & \tau_{\mathsf{stem}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x}' + s_0 \geq 0 & (\varphi_1^{\text{BMS}}) \\
\forall \boldsymbol{x}\, \forall \boldsymbol{x}' & \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0 \;\wedge\; \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x}' + s_0 \geq 0 & (\varphi_2^{\text{BMS}}) \\
\exists \delta > 0\, \forall \boldsymbol{x}\, \forall \boldsymbol{x}' & \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0 \;\wedge\; \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x} - \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x}' \geq \delta & (\varphi_3^{\text{BMS}}) \\
\forall \boldsymbol{x}\, \forall \boldsymbol{x}' & \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0 \;\wedge\; \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x} + r_0 \geq 0 & (\varphi_4^{\text{BMS}})
\end{array}$$

The free variables of $\varphi_1^{\text{BMS}} \wedge \varphi_2^{\text{BMS}} \wedge \varphi_3^{\text{BMS}} \wedge \varphi_4^{\text{BMS}}$ are $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0.

Transformation 1: Move supporting invariant to right-hand side. We bring the conjunct $\boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} + s_0 \geq 0$ in three of the four constraints (φ_1^{BMS}), (φ_2^{BMS}), (φ_3^{BMS}), and (φ_4^{BMS}) to the right-hand side of the implication, according to the following scheme.

$$\phi_1 \wedge \phi_2 \rightarrow \psi \quad \equiv \quad \phi_2 \rightarrow \psi \vee \neg \phi_1$$

[2] In [5] the authors use more general general constraints that can be used to synthesize lexicographic linear ranking functions together with a conjunction of linear supporting invariants for programs that can also contains disjunctions.

We obtain the following constraints.

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{stem}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 \geq 0 \tag{ψ_1}$$

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 \geq 0 \ \vee\ -\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0 \tag{ψ_2}$$

$$\exists \delta > 0\ \forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} - \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x}' \geq \delta \ \vee\ -\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0 \tag{ψ_3}$$

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} + r_0 \geq 0 \ \vee\ -\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0 \tag{ψ_4}$$

Transformation 2: Drop supporting invariant in fourth constraint. We strengthen the fourth constraint (ψ_4) by removing the disjunct $-\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0$. A solution for the strengthened constraint defines a ranking function whose value is bounded from below for all states (and not just those that satisfy the supporting invariant).

Transformation 3: Replace disjunction by sum. We replace the disjunction on the right-hand side of the implication in constraints (ψ_2) and (ψ_3) by a single inequality, according to the scheme below. (It is the disjunction which prevents us from applying Farkas' Lemma to the constraints (ψ_2) and (ψ_3).)

$$m \geq 0 \vee n > 0 \qquad \rightsquigarrow \qquad m + n \geq 0$$

In the second constraint (ψ_2), we replace the disjunction

$$-\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0 \ \vee\ \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 \geq 0$$

by the inequality

$$\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 - \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 \geq 0.$$

In the third constraint (ψ_3), we replace the disjunction

$$-\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 > 0 \ \vee\ \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} - \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x}' \geq \delta$$

by the inequality

$$\boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} - \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x}' - \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 \geq \delta.$$

We obtain the following four constraints.

The Or-to-Plus constraints

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{stem}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 \geq 0 \tag{φ_1}$$

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' + s_0 - \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 \geq 0 \tag{φ_2}$$

$$\exists \delta > 0\ \forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} - \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x}' - \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} - s_0 \geq \delta \tag{φ_3}$$

$$\forall \boldsymbol{x}\ \forall \boldsymbol{x}' \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^{\mathsf{T}} \cdot \boldsymbol{x} + r_0 \geq 0 \tag{φ_4}$$

The free variables of the conjunction $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$ are $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0.

Since we consider linear lasso programs, the relations τ_{stem} and τ_{loop} are given as conjunctions of linear constraints.

$$\begin{aligned}\tau_{\mathsf{stem}}(\boldsymbol{x}, \boldsymbol{x}') &\equiv A_{\mathsf{stem}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{stem}} \\ \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') &\equiv A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}}\end{aligned}$$

We have now finished the description for the three transformation steps that lead us to the the or-to-plus constraints. We are now ready to introduce our method.

The Or-to-Plus Method

Input: linear lasso program P.
Output: coefficients $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0 of a linear ranking function with linear supporting invariant

1. Set up the or-to-Plus constraints φ_1, φ_2, φ_3, and φ_4 for P.
2. Apply Farkas' Lemma to each constraint.
3. Obtain $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0, by linear constraint solving.

After setting up the four or-to-plus constraints φ_1, φ_2, φ_3, φ_4 in Step 1, we apply Farkas' Lemma to each of the four constraints in Step 2. We obtain four linear constraints. E.g., by applying Farkas' Lemma to the constraint (φ_3) we obtain the following linear constraint.

$$\exists \delta > 0 \quad \exists \boldsymbol{\lambda} \quad \boldsymbol{\lambda} \geq 0 \quad \wedge \quad \boldsymbol{\lambda}^{\mathsf{T}} \cdot A_{\mathsf{loop}} = \binom{\boldsymbol{s} - \boldsymbol{r}}{\boldsymbol{r}}^{\mathsf{T}} \quad \wedge \quad \boldsymbol{\lambda}^{\mathsf{T}} \cdot \boldsymbol{b}_{\mathsf{loop}} \leq -\delta - s_0$$

We apply linear constraint solving in Step 3. We obtain a satisfying assignment for the free variables in the resulting constraints. The values obtained for $\boldsymbol{r}$, r_0, $\boldsymbol{s}$ and s_0 are the coefficients of a linear ranking function $f(\boldsymbol{x})$ with linear supporting invariant $I(\boldsymbol{x})$.

The or-to-plus method inherits its soundness from method of Bradley–Manna–Sipma. Step 1 is an equivalence transformation on the Bradley–Manna–Sipma constraints, Step 2 and Step 3 strengthen the constraints, and the application of Farkas' Lemma is an equivalence transformation. Thus, a satisfying assignment of the or-to-plus constraints obtained after the application of Farkas' Lemma is also a satisfying assignment of the Bradley–Manna–Sipma constraints.

5 Completeness of the Or-to-Plus Method

In the tradition of constraint-based synthesis for verification, we will formulate completeness according to the following scheme: the method `X` applied to a program P in the class `Y` computes (the coefficients of) a correctness argument of the form `Z` whenever one exists (i.e., whenever a correctness argument of the form `Z` exists for the program P). Here, `X` is the or-to-plus method, `Y` is the class of lasso programs, and `Z` is a termination argument consisting of a linear ranking function and an invariant of a form that we we define next.

```
x := y + 42;
while( x >= 0 ) {
    y := 2*y - x;
    x := (y + x) / 2;
}
```

$$\tau_{\mathsf{stem}} : x' = y + 42 \ \wedge \ y' = y$$

$$\tau_{\mathsf{loop}} : x \geq 0 \ \wedge \ x' = y \ \wedge \ y' = 2y - x$$

Fig. 2. Linear lasso program $P_{\mathsf{diff42}} = (\tau_{\mathsf{stem}}, \tau_{\mathsf{loop}})$ that has the linear ranking function $f(x, y) = x$ with linear supporting invariant $x - y \geq 42$.

Definition 3 (Non-decreasing linear inductive invariant). *We call a linear inductive invariant* $\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} + s_0 \geq 0$ *of the lasso program* P non-decreasing *if the loop implies that the value of the term* $\boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x} + s_0$ *does not decrease when executing the loop, i.e.,*

$$\tau_{\mathsf{loop}} \rightarrow \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}' \geq \boldsymbol{s}^{\mathsf{T}} \cdot \boldsymbol{x}.$$

In Section 6 we give examples which may help to convey some intuition about the meaning of 'non-decreasing', examples of those terminating programs that do have a linear ranking function with a non-decreasing linear supporting invariant, and examples of those that don't.

Theorem 1 (Completeness). *The or-to-plus method applied to the linear lasso program* P *succeeds and computes the coefficients of a linear ranking function with non-decreasing linear supporting invariant whenever one exists.*

To prove this theorem we use the following lemma.

Lemma 1. *Given are*

(1) satisfiable linear inequalities $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$,
(2) an inequality $\boldsymbol{g}^{\mathsf{T}} \cdot \boldsymbol{x} + g_0 \geq 0$, *and*
(3) a strict inequality $\boldsymbol{h}^{\mathsf{T}} \cdot \boldsymbol{x} + h_0 > 0$.

If $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ *does not imply the strict inequality (3), but the disjunction of (2) and (3), i.e.*

$$\forall \boldsymbol{x} \quad A \cdot \boldsymbol{x} \leq \boldsymbol{b} \ \rightarrow \ \boldsymbol{g}^{\mathsf{T}} \cdot \boldsymbol{x} + g_0 \geq 0 \ \vee \ \boldsymbol{h}^{\mathsf{T}} \cdot \boldsymbol{x} + h_0 > 0,$$

then there exists a constant $\mu \geq 0$ *such that*

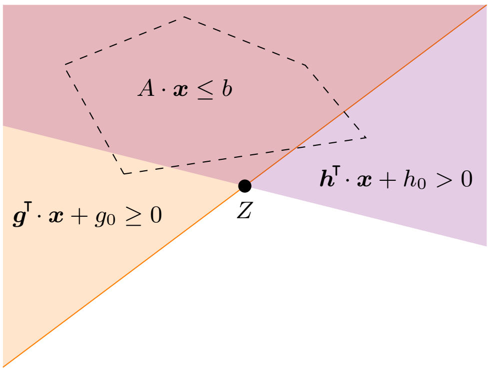
$$\forall \boldsymbol{x} \quad A \cdot \boldsymbol{x} \leq \boldsymbol{b} \ \rightarrow \ (\boldsymbol{g}^{\mathsf{T}} \cdot \boldsymbol{x} + g_0) + \mu \cdot (\boldsymbol{h}^{\mathsf{T}} \cdot \boldsymbol{x} + h_0) \geq 0.$$

Proof (of Lemma 1).

$$\forall \boldsymbol{x} \quad A \cdot \boldsymbol{x} \leq \boldsymbol{b} \ \rightarrow \ (\boldsymbol{g}^{\mathsf{T}} \cdot \boldsymbol{x} + g_0 \geq 0 \ \vee \ \boldsymbol{h}^{\mathsf{T}} \cdot \boldsymbol{x} + h_0 > 0)$$

is equivalent to

$$\forall \boldsymbol{x} \quad (A \cdot \boldsymbol{x} \leq \boldsymbol{b} \ \wedge \ \boldsymbol{h}^{\mathsf{T}} \cdot \boldsymbol{x} + h_0 \leq 0) \ \rightarrow \ \boldsymbol{g}^{\mathsf{T}} \cdot \boldsymbol{x} + g_0 \geq 0.$$



Let $H = \{\boldsymbol{x} \mid \boldsymbol{g}^\mathsf{T} \cdot \boldsymbol{x} + g_0 \geq 0\}$, and $H' = \{\boldsymbol{x} \mid \boldsymbol{h}^\mathsf{T} \cdot \boldsymbol{x} + h_0 > 0\}$ be half-spaces defined by linear inequalities. A half-space $H_\mu = \{\boldsymbol{x} \mid (\boldsymbol{g}^\mathsf{T} \cdot \boldsymbol{x} + g_0) + \mu \cdot (\boldsymbol{h}^\mathsf{T} \cdot \boldsymbol{x} + h_0) \geq 0\}$ defined by a weighted sum is a rotation of H around the intersection Z of the boundary of H and the boundary of H'.

If a polyhedron X is contained in the union $H \cup H'$, then there is a half-space H_μ defined by a weighted sum that contains X.

Fig. 3. A geometrical interpretation of Lemma 1.

By assumption, *(1)* does not imply *(3)*, so $A \cdot \boldsymbol{x} \leq \boldsymbol{b} \wedge \boldsymbol{h}^\mathsf{T} \cdot \boldsymbol{x} + h_0 \leq 0$ is satisfiable, and by Farkas' Lemma this formula is equivalent to

$$\exists \mu \geq 0 \; \exists \boldsymbol{\lambda} \geq 0 \quad \mu \cdot \boldsymbol{h}^\mathsf{T} + \boldsymbol{\lambda}^\mathsf{T} \cdot A = -\boldsymbol{g}^\mathsf{T} \;\wedge\; \boldsymbol{\lambda}^\mathsf{T} \cdot \boldsymbol{b} + \mu \cdot (-h_0) \leq g_0,$$

and thus

$$\exists \mu \geq 0 \; \exists \boldsymbol{\lambda} \geq 0 \quad \boldsymbol{\lambda}^\mathsf{T} \cdot A = -(\mu \cdot \boldsymbol{h}^\mathsf{T} + \boldsymbol{g}^\mathsf{T}) \;\wedge\; \boldsymbol{\lambda}^\mathsf{T} \cdot \boldsymbol{b} \leq \mu \cdot h_0 + g_0.$$

Because $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable by assumption, Farkas' Lemma can be applied again to yield

$$\exists \mu \geq 0 \; \forall \boldsymbol{x} \qquad A \cdot \boldsymbol{x} \leq \boldsymbol{b} \;\rightarrow\; -(\mu \cdot \boldsymbol{h}^\mathsf{T} + \boldsymbol{g}^\mathsf{T})\boldsymbol{x} \leq \mu \cdot h_0 + g_0. \quad \square$$

Proof (of Theorem 1). Let $f(\boldsymbol{x}) = \mathring{\boldsymbol{r}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{r}_0$ be a ranking function with non-decreasing supporting invariant $I(\boldsymbol{x}) \equiv \mathring{\boldsymbol{s}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{s}_0 \geq 0$ for the lasso program P. Since executions of our lasso programs comprise at least three states, there can be no supporting invariant that contradicts the loop, i.e.

$$A_{\mathsf{loop}} \cdot \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; -\mathring{\boldsymbol{s}}^\mathsf{T} \cdot \boldsymbol{x} - \mathring{s}_0 > 0 \tag{1}$$

is not valid. From ($\varphi_{\mathsf{rkBound}}$) it follows that

$$\mathring{\boldsymbol{s}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{s}_0 \geq 0 \;\wedge\; A_{\mathsf{loop}} \cdot \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; \mathring{\boldsymbol{r}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{r}_0 \geq 0,$$

and hence the implication

$$A_{\mathsf{loop}} \cdot \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; \mathring{\boldsymbol{r}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{r}_0 \geq 0 \;\vee\; -\mathring{\boldsymbol{s}}^\mathsf{T} \cdot \boldsymbol{x} - \mathring{s}_0 > 0$$

is valid. By (1) and Lemma 1 there is a $\mu_1 \geq 0$ such that

$$A_{\mathsf{loop}} \cdot \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; (\mathring{\boldsymbol{r}}^\mathsf{T} \cdot \boldsymbol{x} + \mathring{r}_0) \;+\; \mu_1 \cdot (-\mathring{\boldsymbol{s}}^\mathsf{T} \cdot \boldsymbol{x} - \mathring{s}_0) \geq 0$$

is valid. If we assign $\boldsymbol{r} \mapsto \mathring{\boldsymbol{r}} - \mu_1 \cdot \mathring{s}, r_0 \mapsto \mathring{r}_0 - \mu_1 \cdot \mathring{s}_0$, then ($\varphi_4$) is satisfied.

Because $I(\boldsymbol{x}) \equiv \mathring{\boldsymbol{s}} \cdot \boldsymbol{x} + \mathring{s}_0 \geq 0$ is a non-decreasing invariant,

$$A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; \mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x}' - \mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x} \geq 0,$$

and hence, since $\mu_1 \geq 0$,

$$A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; -\mu_1 \cdot \mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot (\boldsymbol{x} - \boldsymbol{x}') \geq 0. \tag{2}$$

From ($\varphi_{\mathsf{rkDecr}}$) we know that

$$\mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x} + \mathring{s}_0 \geq 0 \;\wedge\; A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; \mathring{\boldsymbol{r}}^{\mathsf{T}} \cdot \boldsymbol{x} - \mathring{\boldsymbol{r}}^{\mathsf{T}} \cdot \boldsymbol{x}' \geq \delta,$$

and hence equivalently

$$A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; \mathring{\boldsymbol{r}}^{\mathsf{T}} \cdot \boldsymbol{x} - \mathring{\boldsymbol{r}}^{\mathsf{T}} \cdot \boldsymbol{x}' \geq \delta \;\vee\; -\mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x} - \mathring{s}_0 > 0.$$

With (2) we obtain validity of the following formula.

$$A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; (\mathring{\boldsymbol{r}}^{\mathsf{T}} - \mu_1 \cdot \mathring{\boldsymbol{s}}^{\mathsf{T}}) \cdot (\boldsymbol{x} - \boldsymbol{x}') \geq \delta \;\vee\; -\mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x} - \mathring{s}_0 > 0$$

By (1) and Lemma 1 there exists a $\mu_2 \geq 0$ such that

$$A_{\mathsf{loop}} \cdot \binom{\boldsymbol{x}}{\boldsymbol{x}'} \leq \boldsymbol{b}_{\mathsf{loop}} \;\rightarrow\; (\mathring{\boldsymbol{r}}^{\mathsf{T}} - \mu_1 \cdot \mathring{\boldsymbol{s}}^{\mathsf{T}}) \cdot (\boldsymbol{x} - \boldsymbol{x}') + \mu_2 \cdot (-\mathring{\boldsymbol{s}}^{\mathsf{T}} \cdot \boldsymbol{x} - \mathring{s}_0) > \delta.$$

We pick the assignment $\boldsymbol{r} \mapsto \mathring{\boldsymbol{r}} - \mu_1 \cdot \mathring{s}, r_0 \mapsto \mathring{r}_0 - \mu_1 \cdot \mathring{s}_0, \boldsymbol{s} \mapsto \mu_2 \cdot \mathring{\boldsymbol{s}}, s_0 \mapsto \mu_2 \cdot \mathring{s}_0$, which hence satisfies (φ_3). We already argued that it satisfies (φ_4), and from $\mu_2 \geq 0$ and the fact that $I(\boldsymbol{x})$ is a non-decreasing inductive invariant it follows that the assignment also satisfies (φ_1) and (φ_2). Hence, the ranking function $(\mathring{\boldsymbol{r}} - \mu_1 \cdot \mathring{s})^{\mathsf{T}} \cdot \boldsymbol{x} + \mathring{r}_0 - \mu_1 \cdot \mathring{s}_0$ with supporting invariant $(\mu_2 \cdot \mathring{\boldsymbol{s}})^{\mathsf{T}} \cdot \boldsymbol{x} + \mu_2 \cdot \mathring{s}_0 \geq 0$ can be found by the or-to-plus method. □

6 Examples

```
y := 23;
while( x >= y ) {
    x := x - 1;
}
```

Fig. 4. Lasso program P_{bound}

Our three transformations strengthened the Bradley–Manna–Sipma constraints, hence the solution space of the or-to-plus constraints is smaller than the solution space of the Bradley–Manna–Sipma constraints. This can be seen e.g., in the example depicted in Figure 4. The program P_{bound} has the linear ranking function $f(x, y) = x$ with linear supporting invariant $y \geq 23$, but the coefficients of this ranking function and supporting invariant are no solution of the or-to-plus constraints; the constraint φ_4 is violated. Does this mean that our method will not succeed? No, it does not. By Theorem 1, in fact, we do know that the method will succeed. I.e., since we know of some linear ranking function with non-decreasing supporting invariant (in this case, $f(x, y) = x$ and $y \geq 23$), even if it is not a solution, we know that there exists one which is a solution (here, for example, $f(x, y) = x - y$ with the (trivial) supporting invariant $0 \geq 0$).

```
y := 2;
while( x >= 0 ) {
    x := x - y;
    y := (y + 1) / 2;
}
```

Fig. 5. Lasso program P_{zeno}

The prerequisite of Theorem 1 is the existence of a non-decreasing supporting invariant. There are linear lasso programs that have a linear ranking function with linear supporting invariant, but do not have a linear ranking function with a non-decreasing linear supporting invariant. E.g., for the lasso programs depicted in Figure 5 and Figure 6 our or-to-plus method is not able to synthesize a ranking function for these programs.

The linear lasso program P_{zeno} depicted in Figure 5 has the linear ranking function $f(x, y) = x$ with the linear supporting invariant $y \geq 1$. However this inductive invariant is not non-decreasing; while executing the loop the value of the variable y converges to 1 in the following sequence. $2,\ 1+\frac{1}{2},\ 1+\frac{1}{4},\ 1+\frac{1}{8}, \ldots$.

The statement `havoc y;` in the lasso program P_{wild} is a nondeterministic assignment to the variable y. The relations τ_{stem} and τ_{loop} of this lasso program are given by the constraints $y' \geq 1$ and $x \geq 0 \ \land \ x' = x - y \ \land \ y' \geq 1$. P_{wild} has the ranking function $f(x, y) = x$ with the supporting invariant $y \geq 1$, however this inductive invariant is not non-decreasing in each execution of the loop the variable y can get any value greater than or equal to one.

```
assume y >= 1;
while( x >=0 ) {
    x := x - y;
    havoc y;
    assume (y >= 1);
}
```

Fig. 6. Lasso program P_{wild}

The next example shows that nondeterministic updates are no general obstacle for our or-to-plus method. In the linear lasso program P_{array} the loop iterates over an array of positive integers. The index accessed in the next iteration is the sum of the current index, the current entry of the array, and an offset. The relations τ_{stem} and τ_{loop} of this lasso program are given by the constraints $\mathit{offset}' = 1 \ \land \ i' = 0$ and $i \leq a.\mathit{length} \ \land \ \mathit{curVal}' \geq 0 \ \land \ i' = i + \mathit{offset} + \mathit{curVal}'$. The variable curVal which represents the current entry of the array `a[i]` can get any value greater than or equal to one in each loop iteration. The or-to-plus method finds the linear ranking function $f(i, \mathit{offset}) = i - a.\mathit{length}$ with the linear supporting invariant $\mathit{offset} \geq 1$.

```
offset := 1;
i := 0;
while(i<=a.length) {
    assume a[i]>=0;
    i := i + offset + a[i];
}
```

Fig. 7. Lasso program P_{array}

7 Lasso Programs over the Integers

In the preceding sections we considered lasso programs over the reals. In this section we discuss the applicability of the or-to-plus method to linear lasso programs over the integers, i.e., programs where the set of states Σ is a subset of $\mathbb{Z}^n$. We still use real-valued ranking functions. We obtain the constraints for coefficients of a linear ranking function with linear supporting invariant by restricting the range of the universal quantification in the constraints φ_1, φ_2, φ_3, and φ_4 to the integers. E.g., the constraint φ_3 for linear lasso programs over the integers is

$$\exists \delta > 0\ \forall \boldsymbol{x} \in \mathbb{Z}^n\ \forall \boldsymbol{x}' \in \mathbb{Z}^n \quad \tau_{\mathsf{loop}}(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x} - \boldsymbol{r}^\mathsf{T} \cdot \boldsymbol{x}' - \boldsymbol{s}^\mathsf{T} \cdot \boldsymbol{x} - s_0 \geq \delta$$

where the domain of the coefficients $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0 and the quantified variable δ are the reals. Now, Farkas' lemma is not an equivalence transformation, its application results in weaker formulas. This means the or-to-plus method is still sound, but we loose the completeness result of Theorem 1. An example for this is the program $P_{\mathsf{nonIntegral}}$, depicted in Figure 8 that has the following transition relations.

```
assume 2*y >= 1;
while( x >= 0 ) {
    x := x - 2*y + 1;
}
```

Fig. 8. Lasso program $P_{\mathsf{nonIntegral1}}$

$$\tau_{\mathsf{stem}} : 2y' \geq 1 \ \wedge\ x' = x$$

$$\tau_{\mathsf{loop}} : x \geq 0 \ \wedge\ x' = x - 2y + 1 \ \wedge\ y' = y;$$

Over integer variables, $P_{\mathsf{nonIntegral1}}$ has the linear ranking function $f(x, y) = x$ with the linear supporting invariant $y \geq 1$. Over real-valued variables, $P_{\mathsf{nonIntegral1}}$ does not terminate. If we add the additional constraint $y' \geq 1$ to τ_{stem}, the programs' semantics over the integers is not changed, but we are able to synthesize a linear ranking function with a linear supporting invariant. Adding this additional constraint gives the constraints a property that we formally define as follows.

Integral constraints. A conjunction of linear constraints $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ is called *integral* if the set of satisfying assignments over the reals $S := \{\boldsymbol{r} \in \mathbb{R}^n \mid A \cdot \boldsymbol{r} \leq \boldsymbol{b}\}$ coincides with the integer hull of S (the convex hull of all integer vectors in S).

For each conjunction of m linear constraints there is an equivalent conjunction of at most 2^m linear constraints that is integral [24]. We add an additional step to the or-to-plus method in which we make the constraints in the stem transition τ_{stem} and loop transition τ_{loop} integral.

The Or-to-Plus Method (Int)

Input: linear lasso program P with integer variables
Output: coefficients $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0 of linear ranking function with linear supporting invariant

1. Replace τ_{stem} and τ_{loop} by equivalent integral linear constraints.
2. Set up constraints φ_1, φ_2, φ_3, and φ_4 for P.
3. Apply Farkas' Lemma to each constraint.
4. Obtain $\boldsymbol{r}$, r_0, $\boldsymbol{s}$, and s_0, by linear constraint solving.

That we find more solutions after making the linear constraints τ_{stem} and τ_{loop} integral is due to the following lemma which was stated in [12]. We present our proof for the purpose of self-containment.

Lemma 2 (Integral version of Farkas' Lemma). *Given a conjunction of linear constraints $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ which is satisfiable and integral, and a linear constraint $\boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \leq \delta$,*

$$\forall \boldsymbol{x} \in \mathbb{Z}^n \;\; (A \cdot \boldsymbol{x} \leq \boldsymbol{b} \rightarrow \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \leq \delta) \quad \textit{iff} \quad \exists \boldsymbol{\lambda} \;\; (\boldsymbol{\lambda} \geq 0 \wedge \boldsymbol{\lambda}^\mathsf{T} \cdot A = \boldsymbol{c}^\mathsf{T} \wedge \boldsymbol{\lambda}^\mathsf{T} \cdot \boldsymbol{b} \leq \delta)$$

Proof. We write this statement as a linear programming problem.

$$\textbf{(P)} \qquad \max\{\boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \mid A \cdot \boldsymbol{x} \leq \boldsymbol{b}\} \leq \delta$$

Because the constraints $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ are integral, there is an integral vector $\boldsymbol{x} \in \mathbb{Z}^n$ such that $\boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x}$ is the optimum solution to **(P)**. Thus the optimum over integers is $\leq \delta$ if and only if the optimum of the reals is. The statement now follows from the real version of Farkas' Lemma. □

However, even if τ_{stem} and τ_{loop} are integral, our method is not complete over the integers. In the completeness proof for the reals we applied Farkas' Lemma to conjunctions of a polyhedron $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$ and an inequality $\boldsymbol{h}^\mathsf{T} \cdot x + h_0 \leq 0$. This inequality contains free variables, namely the coefficients of the supporting invariant. Even if τ_{stem} and τ_{loop} are integral, this conjunction might not be integral and we cannot apply the integer version of Farkas' lemma in this case.

```
assume 2*y >= z;
while( x >= 0 && z == 1 ) {
    x := x - 2*y + 1;
}
```

Fig. 9. Lasso program $P_{\mathsf{nonIntegral2}}$

A counterexample to completeness of our integer version of the or-to-plus method is the linear lasso program $P_{\mathsf{nonIntegral2}}$ depicted in Figure 9.

8 Conclusion

We have presented a constraint-based synthesis method for a class of programs that was not investigated before for the synthesis problem. The class is restricted (though less restricted than the widely studied class of simple while programs) but still requires the combined synthesis of not only a ranking function but also an invariant. We have formulated and proven a completeness theorem that gives us an indication on the extent of power of a method that does without nonlinear constraint solving.

We implemented the or-to-plus method as plugin of the Ultimate software analysis framework. A version that allows one to 'play around' with lasso programs is available via a web interface at the following URL.

`http://ultimate.informatik.uni-freiburg.de/LassoRanker`

As mentioned in the introduction, the class of lasso programs is motivated by the fact that they are a natural way (and, it seems, the only way) to represent an (infinite) counterexample path in a control flow graph. It is a topic of future research to explore the different scenarios in practical tools that use a module to find a ranking function and a supporting invariant for a lasso program (e.g., in [1,13,15,16,17,21,22]) and to compare the performance of our—theoretically motivated—synthesis method in comparison with the existing—heuristically motivated—approach used presently in the module.

References

1. E. Albert, P. Arenas, S. Genaim, and G. Puebla. Closed-form upper bounds in static cost analysis. *J. Autom. Reasoning*, 46(2):161–203, 2011.
2. R. Bagnara, F. Mesnard, A. Pescetti, and E. Zaffanella. A new look at the automatic synthesis of linear ranking functions. *Inf. Comput.*, 215:47–67, 2012.
3. A. M. Ben-Amram and S. Genaim. On the linear ranking problem for integer linear-constraint loops. In *POPL*, 2013.
4. A. M. Ben-Amram, S. Genaim, and A. N. Masud. On the termination of integer loops. In *VMCAI*, pages 72–87, 2012.
5. A. R. Bradley, Z. Manna, and H. B. Sipma. Linear ranking with reachability. In *CAV*, pages 491–504, 2005.
6. A. R. Bradley, Z. Manna, and H. B. Sipma. The polyranking principle. In *ICALP*, pages 1349–1361, 2005.
7. A. R. Bradley, Z. Manna, and H. B. Sipma. Termination analysis of integer linear loops. In *CONCUR*, pages 488–502, 2005.
8. M. Braverman. Termination of integer linear programs. In *CAV*, pages 372–385, 2006.
9. M. Brockschmidt, R. Musiol, C. Otto, and J. Giesl. Automated termination proofs for Java programs with cyclic data. In *CAV*, pages 105–122, 2012.
10. M. Colón and H. Sipma. Synthesis of linear ranking functions. In *TACAS*, pages 67–81, 2001.
11. B. Cook, J. Fisher, E. Krepska, and N. Piterman. Proving stabilization of biological systems. In *VMCAI*, pages 134–149, 2011.
12. B. Cook, D. Kroening, P. Rümmer, and C. M. Wintersteiger. Ranking function synthesis for bit-vector relations. *Formal Methods in System Design*, 2013.
13. B. Cook, A. Podelski, and A. Rybalchenko. Terminator: Beyond safety. In *CAV*, pages 415–418, 2006.
14. P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *VMCAI*, pages 1–24, 2005.
15. S. Gulwani and F. Zuleger. The reachability-bound problem. In B. G. Zorn and A. Aiken, editors, *PLDI*, pages 292–304. ACM, 2010.
16. A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, and R.-G. Xu. Proving non-termination. In *POPL*, pages 147–158, 2008.
17. W. R. Harris, A. Lal, A. V. Nori, and S. K. Rajamani. Alternation for termination. In *SAS*, pages 304–319, 2010.
18. D. Kroening, N. Sharygina, S. Tonetta, A. Tsitovich, and C. M. Wintersteiger. Loop summarization using abstract transformers. In *ATVA*, pages 111–125, 2008.
19. D. Kroening, N. Sharygina, A. Tsitovich, and C. M. Wintersteiger. Termination analysis with compositional transition invariants. In *CAV*, pages 89–103, 2010.
20. A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, pages 239–251, 2004.
21. A. Podelski and A. Rybalchenko. Transition invariants. In *LICS*, pages 32–41, 2004.

22. A. Podelski and S. Wagner. A sound and complete proof rule for region stability of hybrid systems. In *HSCC*, pages 750–753, 2007.
23. A. Rybalchenko. Constraint solving for program verification: Theory and practice by example. In *CAV*, pages 57–71, 2010.
24. A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.
25. A. Tiwari. Termination of linear programs. In *CAV*, pages 70–82, 2004.