arXiv:1609.05207v1 [cs.LO] 16 Sep 2016

Geometric Nontermination Arguments

Jan Leike[1] and Matthias Heizmann[2]

[1] Australian National University
[2] University of Freiburg

Abstract. We present a new kind of nontermination argument, called *geometric nontermination argument*. The geometric nontermination argument is a finite representation of an infinite execution that has the form of a sum of several geometric series. For so-called linear lasso programs we can decide the existence of a geometric nontermination argument using a nonlinear algebraic ∃-constraint. We show that a deterministic conjunctive loop program with nonnegative eigenvalues is nonterminating if an only if there exists a geometric nontermination argument. Furthermore, we present an evaluation that demonstrates that our method is feasible in practice.

1 Introduction

The problem whether a program is terminating is undecidable in general. One way to approach this problem in practice is to analyze the existence of termination arguments and nontermination arguments. The existence of a certain termination argument like, e.g, a linear ranking function, is decidable [28,3] and implies termination. However, if we cannot find a linear ranking function we cannot conclude nontermination. Vice versa, the existence of a certain nontermination argument like, e.g, a linear recurrence set [18], is decidable and implies nontermination however, if we cannot find such a recurrence set we cannot conclude termination.

In this paper we present a new kind of termination argument which we call *geometric nontermination argument (GNTA)*. Unlike a recurrence set, a geometric nontermination argument does not only imply nontermination, it also explicitly represents an infinite program execution. An infinite program execution that is represented by a geometric nontermination argument can be written as a pointwise sum of several geometric series. We show that such an infinite execution exists for each deterministic conjunctive loop program that is nonterminating and whose transition matrix has only nonnegative eigenvalues.

We restrict ourselves to linear lasso programs. A lasso program consists of a single while loop that is preceded by straight-line code. The name refers to the lasso shaped form of the control flow graph. 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 [14,6,19,22,23,29,30,20], stability analysis [10,31], cost analysis [1,17], or the verification of temporal properties [13,12,16] for programs.

```
b := 1;
while (a+b >= 3):
  a := 3*a + 1;
  b := nondet();
```

(a)

```
b := 1;
while (a+b >= 3):
  a := 3*a - 2;
  b := 2*b;
```

(b)

```
b := 1;
while (a+b >= 4):
  a := 3*a + b;
  b := 2*b;
```

(c)

Fig. 1: Three nonterminating linear lasso programs. Each has an infinite execution which is either a geometric series or a pointwise sum of geometric series. The first lasso program is nondeterministic because the variable `b` gets some nondeterministic value in each iteration.

We present a constraint based approach that allow us to check whether a linear conjunctive lasso program has a geometric nontermination argument and to synthesize one if it exists.

Our analysis is motived by the probably simplest form of an infinite executions, namely infinite execution where the same state is always repeated. We call such a state a fixed point. For lasso programs we can reduce the check for the existence of a fixed point to a constraint solving problem as follows. Let us assume that the stem and the loop of the lasso program are given as a formulas over primed and unprimed variables $\textsc{stem}(\boldsymbol{x}, \boldsymbol{x}')$ and $\textsc{loop}(\boldsymbol{x}, \boldsymbol{x}')$. The infinite sequence $\boldsymbol{s}_0, \bar{\boldsymbol{s}}, \bar{\boldsymbol{s}}, \bar{\boldsymbol{s}}, \ldots$ is an nonterminating execution of the lasso program iff the assignment $\boldsymbol{x}_0 \mapsto \boldsymbol{s}_0, \bar{\boldsymbol{x}} \mapsto \bar{\boldsymbol{s}}$ is a satisfying assignment for the constraint $\textsc{stem}(\boldsymbol{x}_0, \bar{\boldsymbol{x}}) \wedge \textsc{loop}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}})$. In this paper, we present a constraint that is not only satisfiable if the program has a fixed point, it is also satisfiable if the program has a nonterminating execution that can be written as a pointwise sum of geometric series.

Let us motivate the representation of infinite executions as sums of geometric series in three steps. The program depicted in Figure 1a shows a lasso program which does not have a fixed point but the following infinite execution.

$$\left(\begin{smallmatrix}2\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}2\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}7\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}22\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}67\\1\end{smallmatrix}\right), \ldots$$

We can write this infinite execution as a a geometric series where for $k > 1$ the k-th state is the sum $\boldsymbol{x_1} + \sum_{i=0}^{k} \lambda^i \boldsymbol{y}$, where we have $\boldsymbol{x_1} = \left(\begin{smallmatrix}2\\1\end{smallmatrix}\right)$, $\boldsymbol{y} = \left(\begin{smallmatrix}5\\0\end{smallmatrix}\right)$, and $\lambda = 2$. The state $\boldsymbol{x_1}$ is the state before the loop was executed before the first time and intuitively $\boldsymbol{y}$ is the direction in which the execution is moving initially and λ is the speed at which the execution continues to move in this direction.

Next, let us consider the lasso program depicted in Figure 1b which has the following infinite execution.

$$\left(\begin{smallmatrix}2\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}2\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}4\\4\end{smallmatrix}\right), \left(\begin{smallmatrix}10\\8\end{smallmatrix}\right), \left(\begin{smallmatrix}28\\16\end{smallmatrix}\right), \ldots$$

We cannot write this execution as a geometric series as we did above. Intuitively, the reason is that the values of both variables are increasing at different speeds

and hence this execution is not moving in a single direction. However, we can write this infinite execution as a sum of geometric series where for $k > 1$ the k-th state can be written as a sum $\boldsymbol{x_1} + \sum_{i=0}^{k} Y \left(\begin{smallmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{smallmatrix}\right)^i \mathbf{1}$, where we have $\boldsymbol{x_1} = \left(\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right)$, $\boldsymbol{Y} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$, $\lambda_1 = 3, \lambda_2 = 2$ and $\mathbf{1}$ denotes the column vector of ones. Intuitively, our execution is moving in two different directions at different speeds. The directions are reflected by the column vectors of Y, the values of λ_1 and λ_2 reflect the respective speeds.

Let us next consider the lasso program in Figure 1c which has the following infinite execution.

$$\left(\begin{smallmatrix} 3 \\ 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 3 \\ 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 10 \\ 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 32 \\ 4 \end{smallmatrix}\right), \left(\begin{smallmatrix} 100 \\ 8 \end{smallmatrix}\right), \ldots$$

We cannot write this execution as a pointwise sum of geometric series in the form that we used above. Intuitively, the problem is that one of the initial directions contributes at two different speeds to the overall progress of the execution. However, we can write this infinite execution as a pointwise sum of geometric series where for $k > 1$ the k-th state can be written as a sum $\boldsymbol{x_1} + \sum_{i=0}^{k} Y \left(\begin{smallmatrix} \lambda_1 & \mu \\ 0 & \lambda_2 \end{smallmatrix}\right)^i \mathbf{1}$, where we have $\boldsymbol{x_1} = \left(\begin{smallmatrix} 3 \\ 1 \end{smallmatrix}\right)$, $\boldsymbol{Y} = \begin{pmatrix} 4 & 3 \\ 0 & 1 \end{pmatrix}$, $\lambda_1 = 3, \lambda_2 = 2, \mu = 1$ and $\mathbf{1}$ denotes the column vector of ones. We call the tuple $(\boldsymbol{x_0}, \boldsymbol{x_1}, Y, \lambda_1, \lambda_2, \mu)$ which we use as a finite representation for the infinite execution a *geometric nontermination argument*.

In this paper, we formally introduce the notion of a geometric nontermination argument for linear lasso programs (Section 3) and we prove that each nonterminating deterministic conjunctive linear loop program whose transition matrix has only nonnegative real eigenvalues has a geometric nontermination argument, i.e., each such nonterminating linear loop program has an infinite execution which can be written as a sum of geometric series (Section 4).

2 Preliminaries

We denote vectors $\boldsymbol{x}$ with bold symbols and matrices with uppercase Latin letters. Vectors are always understood to be column vectors, $\mathbf{1}$ denotes a vector of ones, $\mathbf{0}$ denotes a vector of zeros (of the appropriate dimension), and $\boldsymbol{e_i}$ denotes the i-th unit vector. A list of notation can be found on page 18.

2.1 Linear Lasso Programs

In this work, we consider linear lasso programs, programs that consist of a program step and a single loop. We use binary relations over the program's states to define the stem and the loop transition relation. Variables are assumed to be real-valued.

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

Definition 1 (Linear Lasso Program). *A (conjunctive)* linear lasso program $L = (\textsc{stem}, \textsc{loop})$ *consists of two binary relations defined by formulas with the free variables* $\boldsymbol{x}$ *and* $\boldsymbol{x'}$ *of the form*

$$A \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x'} \end{smallmatrix} \right) \leq \boldsymbol{b}$$

for some matrix $A \in \mathbb{R}^{n \times m}$ *and some vector* $\boldsymbol{b} \in \mathbb{R}^m$.

A *linear loop program* is a linear lasso program L without stem, i.e., a linear lasso program such that the relation STEM is equivalent to *true*.

Definition 2 (Deterministic Linear Lasso Program). *A linear loop program* L *is called* deterministic *iff its loop transition* LOOP *can be written in the following form*

$$(\boldsymbol{x}, \boldsymbol{x'}) \in \textsc{loop} \iff G\boldsymbol{x} \leq \boldsymbol{g} \ \wedge \ \boldsymbol{x'} = M\boldsymbol{x} + \boldsymbol{m}$$

for some matrices $G \in \mathbb{R}^{n \times m}$, $M \in \mathbb{R}^{n \times n}$, *and vectors* $\boldsymbol{g} \in \mathbb{R}^m$ *and* $\boldsymbol{m} \in \mathbb{R}^n$.

Definition 3 (Nontermination). *A linear lasso program* L *is* nonterminating *iff there is an infinite sequence of states* $\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots$, *called an* infinite execution of L, *such that* $(\boldsymbol{x_0}, \boldsymbol{x_1}) \in \textsc{stem}$ *and* $(\boldsymbol{x_t}, \boldsymbol{x_{t+1}}) \in \textsc{loop}$ *for all* $t \geq 1$.

2.2 Jordan Normal Form

Let $M \in \mathbb{R}^{n \times n}$ be a real square matrix. If there is an invertible square matrix S and a diagonal matrix D such that $M = SDS^{-1}$, then M is called *diagonalizable*. The column vectors of S form the basis over which M has diagonal form. In general, real matrices are not diagonalizable. However, every real square matrix M with real eigenvalues has a representation which is almost diagonal, called *Jordan normal form*. This is a matrix that is zero except for the eigenvalues on the diagonal and one superdiagonal containing ones and zeros.

Formally, a Jordan normal form is a matrix $J = \operatorname{diag}(J_{i_1}(\lambda_1), \ldots, J_{i_k}(\lambda_k))$ where $\lambda_1, \ldots, \lambda_k$ are the eigenvalues of M and the real square matrices $J_i(\lambda) \in \mathbb{R}^{i \times i}$ are *Jordan blocks*,

$$J_i(\lambda) := \begin{pmatrix} \lambda & 1 & 0 & \ldots & 0 & 0 \\ 0 & \lambda & 1 & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & \lambda & 1 \\ 0 & 0 & 0 & \ldots & 0 & \lambda \end{pmatrix}.$$

The subspace corresponding to each distinct eigenvalue is called *generalized eigenspace* and their basis vectors *generalized eigenvectors*.

Theorem 4 (Jordan Normal Form). *For each real square matrix* $M \in \mathbb{R}^{n \times n}$ *with real eigenvalues, there is an invertible real square matrix* $V \in \mathbb{R}^{n \times n}$ *and a Jordan normal form* $J \in \mathbb{R}^{n \times n}$ *such that* $M = VJV^{-1}$.

3 Geometric Nontermination Arguments

Fix a conjunctive linear lasso program $L = (\textsc{stem}, \textsc{loop})$ and let $A \in \mathbb{R}^{n \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$ define the loop transition such that

$$(\boldsymbol{x}, \boldsymbol{x}') \in \textsc{loop} \iff A \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix} \right) \leq \boldsymbol{b}.$$

Definition 5 (Geometric Nontermination Argument). *A tuple* $(\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_k}, \lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_{k-1})$ *is called a* geometric nontermination argument *for the linear lasso program* $L = (\textsc{stem}, \textsc{loop})$ *iff all of the following statements hold.*

(domain) $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_k} \in \mathbb{R}^n$, *and* $\lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_{k-1} \geq 0$

(initiation) $(\boldsymbol{x_0}, \boldsymbol{x_1}) \in \textsc{stem}$

(point) $A \left(\begin{smallmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_1} + \sum_i \boldsymbol{y_i} \end{smallmatrix} \right) \leq \boldsymbol{b}$

(ray) $A \left(\begin{smallmatrix} \boldsymbol{y_1} \\ \lambda_1 \boldsymbol{y_1} \end{smallmatrix} \right) \leq 0$ *and* $A \left(\begin{smallmatrix} \boldsymbol{y_i} \\ \lambda_i \boldsymbol{y_i} + \mu_{i-1} \boldsymbol{y_{i-1}} \end{smallmatrix} \right) \leq 0$ *for* $1 < i \leq k$.

The number $k \geq 0$ *is the* size *of the geometric nontermination argument.*

The existence of a geometric nontermination argument can be checked using an SMT solver. The constraints given by (domain), (init), (point), (ray) are nonlinear algebraic constraints. The satisfiability of these constraints is decidable. Moreover, if the linear lasso program is given as a deterministic update, we can compute its eigenvalues. If the eigenvalues are known, we can assign values to $\lambda_1, \ldots, \lambda_k$ and the constraints become linear and can thus be decided efficiently.

Proposition 6 (Soundness). *If there is a geometric nontermination argument for a linear lasso program* L*, then* L *is nonterminating.*

Proof. Define $Y := (\boldsymbol{y_1} \ldots \boldsymbol{y_k})$ denote the matrix containing the vectors $\boldsymbol{y_i}$ as columns, and define the matrix

$$U := \begin{pmatrix} \lambda_1 & \mu_1 & 0 & \ldots & 0 & 0 \\ 0 & \lambda_2 & \mu_2 & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & \lambda_{n-1} & \mu_{n-1} \\ 0 & 0 & 0 & \ldots & 0 & \lambda_n \end{pmatrix}. \tag{1}$$

Following Definition 3 we show that the linear lasso program L has the infinite execution

$$\boldsymbol{x_0},\ \boldsymbol{x_1},\ \boldsymbol{x_1} + Y\mathbf{1},\ \boldsymbol{x_1} + Y\mathbf{1} + YU\mathbf{1},\ \boldsymbol{x_1} + Y\mathbf{1} + YU\mathbf{1} + YU^2\mathbf{1}, \ldots \tag{2}$$

From (init) we get $(\boldsymbol{x_0}, \boldsymbol{x_1}) \in \textsc{stem}$. It remains to show that

$$\left(\boldsymbol{x_1} + \sum_{j=0}^{t-1} YU^j\mathbf{1},\ \boldsymbol{x_1} + \sum_{j=0}^{t} YU^j\mathbf{1} \right) \in \textsc{loop} \text{ for all } t \in \mathbb{N}. \tag{3}$$

According to (domain) the matrix U has only nonnegative entries, so the same holds for the matrix $Z := \sum_{j=0}^{t-1} U^j$. Hence $Z\mathbf{1}$ has only nonnegative entries and thus $YZ\mathbf{1}$ can be written as $\sum_{i=1}^{k} \alpha_i \boldsymbol{y_i}$ for some $\alpha_i \geq 0$. We multiply the inequality number i from (ray) with α_i and get

$$A \left(\begin{smallmatrix} \alpha_i \boldsymbol{y_i} \\ \alpha_i \lambda_i \boldsymbol{y_i} + \alpha_i \mu_{i-1} \boldsymbol{y_{i-1}} \end{smallmatrix} \right) \leq 0. \tag{4}$$

where we use the convenience notation $\boldsymbol{y_0} := 0$ and $\mu_0 := 0$. Now we sum (4) for all i and add (point) to get

$$A \left(\begin{smallmatrix} \boldsymbol{x_1} + \sum_i \alpha_i \boldsymbol{y_i} \\ \boldsymbol{x_1} + \sum_i \boldsymbol{y_i} + \sum_i (\alpha_i \lambda_i \boldsymbol{y_i} + \alpha_i \mu_{i-1} \boldsymbol{y_{i-1}}) \end{smallmatrix} \right) \leq \boldsymbol{b}. \tag{5}$$

By definition of α_i, we have

$$\boldsymbol{x_1} + \sum_{i=1}^{k} \alpha_i \boldsymbol{y_i} \;=\; \boldsymbol{x_1} + YZ\mathbf{1} \;=\; \boldsymbol{x_1} + \sum_{j=0}^{t-1} YU^j \mathbf{1}$$

and

$$\begin{aligned} \boldsymbol{x_1} + \sum_{i=1}^{k} \boldsymbol{y_i} + \sum_{i=1}^{k} (\alpha_i \lambda_i \boldsymbol{y_i} + \alpha_i \mu_{i-1} \boldsymbol{y_{i-1}}) &= \boldsymbol{x_1} + Y\mathbf{1} + \sum_{i=1}^{k} \alpha_i YUe_i \\ &= \boldsymbol{x_1} + Y\mathbf{1} + YUZ\mathbf{1} \\ &= \boldsymbol{x_1} + \sum_{j=0}^{t} YU^j \mathbf{1}. \end{aligned}$$

Therefore (3) and (5) are the same, which concludes this proof. □

Example 7 (Closed Form of the Infinite Execution). The following is the closed form of the state $\boldsymbol{x_t} = \sum_{k=0}^{t} YU^k \mathbf{1}$ in the infinite execution (2). Let $U =: N + D$ where N is a nilpotent matrix and D is a diagonal matrix.

$$YU^k \mathbf{1} = Y \left(\sum_{i=0}^{n} \binom{k}{i} N^i D^{k-i} \right) \mathbf{1} = \sum_{j=1}^{n} y_j \sum_{i=0}^{n-j+1} \binom{k}{i} \lambda_{n-j-i}^{k-i} \prod_{\ell=j}^{j+i-1} \mu_\ell \qquad \Diamond$$

4 Completeness

First we show that a linear loop program has a GNTA if it has is a bounded infinite execution. In the next section we use this to prove our completeness result.

4.1 Bounded Infinite Executions

Let $|\cdot| : \mathbb{R}^n \to \mathbb{R}$ denote some norm. We call an infinite execution $(\boldsymbol{x}_t)_{t \geq 0}$ *bounded* iff there is a real number $d \in \mathbb{R}$ such that the norm of each state is bounded by d, i.e., $|\boldsymbol{x}_t| \leq d$ for all t (in $\mathbb{R}^n$ the notion of boundedness is independent of the choice of the norm).

Lemma 8 (Fixed Point). *Let $L = (\mathit{true}, \textsc{loop})$ be a linear loop program. The linear loop program L has a bounded infinite execution if and only if there is a fixed point $\boldsymbol{x}^* \in \mathbb{R}^n$ such that $(\boldsymbol{x}^*, \boldsymbol{x}^*) \in \textsc{loop}$.*

Proof. If there is a fixed point $\boldsymbol{x}^*$, then the loop has the infinite bounded execution $\boldsymbol{x}^*, \boldsymbol{x}^*, \ldots$. Conversely, let $(\boldsymbol{x}_t)_{t \geq 0}$ be an infinite bounded execution. Boundedness implies that there is an $d \in \mathbb{R}$ such that $|\boldsymbol{x}_t| \leq d$ for all t. Consider the sequence $\boldsymbol{z}_k := \frac{1}{k} \sum_{t=1}^{k} \boldsymbol{x}_t$.

$$\begin{aligned}
|\boldsymbol{z}_k - \boldsymbol{z}_{k+1}| &= \left| \frac{1}{k} \sum_{t=1}^{k} \boldsymbol{x}_t - \frac{1}{k+1} \sum_{t=1}^{k+1} \boldsymbol{x}_t \right| = \frac{1}{k(k+1)} \left| (k+1) \sum_{t=1}^{k} \boldsymbol{x}_t - k \sum_{t=1}^{k+1} \boldsymbol{x}_t \right| \\
&= \frac{1}{k(k+1)} \left| \sum_{t=1}^{k} \boldsymbol{x}_t - k\boldsymbol{x}_{k+1} \right| \leq \frac{1}{k(k+1)} \left(\sum_{t=1}^{k} |\boldsymbol{x}_t| + k|\boldsymbol{x}_{k+1}| \right) \\
&\leq \frac{1}{k(k+1)} (k \cdot d + k \cdot d) = \frac{2d}{k+1} \longrightarrow 0 \text{ as } k \to \infty.
\end{aligned}$$

Hence the sequence $(\boldsymbol{z}_k)_{k \geq 1}$ is a Cauchy sequence and thus converges to some $\boldsymbol{z}^* \in \mathbb{R}^n$. We will show that $\boldsymbol{z}^*$ is the desired fixed point.

For all t, the polyhedron $Q := \{ \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \mid A \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{smallmatrix}\right) \leq b \}$ contains $\left(\begin{smallmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{smallmatrix}\right)$ and is convex. Therefore for all $k \geq 1$,

$$\frac{1}{k} \sum_{t=1}^{k} \left(\begin{smallmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{smallmatrix}\right) \in Q.$$

Together with

$$\left(\begin{smallmatrix} \boldsymbol{z}_k \\ \frac{k+1}{k} \boldsymbol{z}_{k+1} \end{smallmatrix}\right) = \frac{1}{k} \left(\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{x}_1 \end{smallmatrix}\right) + \frac{1}{k} \sum_{t=1}^{k} \left(\begin{smallmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{smallmatrix}\right)$$

we infer

$$\left(\left(\begin{smallmatrix} \boldsymbol{z}_k \\ \frac{k+1}{k} \boldsymbol{z}_{k+1} \end{smallmatrix}\right) - \frac{1}{k} \left(\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{x}_1 \end{smallmatrix}\right) \right) \in Q,$$

and since Q is topologically closed we have

$$\left(\begin{smallmatrix} \boldsymbol{z}^* \\ \boldsymbol{z}^* \end{smallmatrix}\right) = \lim_{k \to \infty} \left(\left(\begin{smallmatrix} \boldsymbol{z}_k \\ \frac{k+1}{k} \boldsymbol{z}_{k+1} \end{smallmatrix}\right) - \frac{1}{k} \left(\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{x}_1 \end{smallmatrix}\right) \right) \in Q. \qquad \square$$

Note that Lemma 8 does not transfer to lasso programs: there might only be one fixed point and the stem might exclude this point (e.g., $a = -0.5$ and $b = 3.5$ in example Figure 1a).

Because fixed points give rise to trivial geometric nontermination arguments, we can derive a criterion for the existence of geometric nontermination arguments from Lemma 8.

Corollary 9 (Bounded Infinite Executions). *If the linear loop program $L = (\mathit{true}, \textsc{loop})$ has a bounded infinite execution, then it has a geometric nontermination argument of size* 0.

Proof. By Lemma 8 there is a fixed point $\boldsymbol{x}^*$ such that $(\boldsymbol{x}^*, \boldsymbol{x}^*) \in \text{LOOP}$. We choose $\boldsymbol{x_0} = \boldsymbol{x_1} = \boldsymbol{x}^*$ which satisfies (point) and (ray) and thus is a geometric nontermination argument for L. □

Example 10. Note that according to our definition of a linear lasso program, the relation LOOP is a topologically closed set. If we allowed the formula defining LOOP to also contain strict inequalities, Lemma 8 no longer holds: the following program is nonterminating and has a bounded infinite execution, but it does not have a fixed point. However, the topological closure of the relation LOOP contains the fixed point $a = 0$.

```
while (a > 0):
    a := a / 2;
```

Nevertheless, this example still has the geometric nontermination argument $\boldsymbol{x_1} = 1$, $\boldsymbol{y_1} = -0.5$, $\lambda = 0.5$. ◊

4.2 Nonnegative Eigenvalues

This section is dedicated to the proof of the following completeness result for deterministic linear loop programs.

Theorem 11 (Completeness). *If a deterministic linear loop program L of the form* `while` *$(G\boldsymbol{x} \leq \boldsymbol{g})$* `do` *$\boldsymbol{x} := M\boldsymbol{x} + \boldsymbol{m}$ with n variables is nonterminating and M has only nonnegative real eigenvalues, then there is a geometric nontermination argument for L of size at most n.*

To prove this completeness theorem, we need to construct a GNTA from a given infinite execution. The following lemma shows that we can restrict our construction to exclude all linear subspaces that have a bounded execution.

Lemma 12 (Loop Disassembly). *Let $L = (\mathit{true}, \text{LOOP})$ be a linear loop program over $\mathbb{R}^n = \mathcal{U} \oplus \mathcal{V}$ where $\mathcal{U}$ and $\mathcal{V}$ are linear subspaces of $\mathbb{R}^n$. Suppose L is nonterminating and there is an infinite execution that is bounded when projected to the subspace $\mathcal{U}$. Let $\boldsymbol{x}^{\mathcal{U}}$ be the fixed point in $\mathcal{U}$ that exists according to Lemma 8. Then the linear loop program $L^{\mathcal{V}}$ that we get by projecting to the subspace $\mathcal{V} + \boldsymbol{x}^{\mathcal{U}}$ is nonterminating. Moreover, if $L^{\mathcal{V}}$ has a GNTA of size k, then L has a GNTA of size k.*

Proof. Without loss of generality, we are in the basis of $\mathcal{U}$ and $\mathcal{V}$ so that these spaces are nicely separated by the use of different variables. Using the infinite execution of L that is bounded on $\mathcal{U}$ we can do the construction from the proof of Lemma 8 to get an infinite execution $\boldsymbol{z_0}, \boldsymbol{z_1}, \ldots$ that yields the fixed point $\boldsymbol{x}^{\mathcal{U}}$ when projected to $\mathcal{U}$. We fix $\boldsymbol{x}^{\mathcal{U}}$ in the loop transition by replacing all variables from $\mathcal{U}$ with the values from $\boldsymbol{x}^{\mathcal{U}}$ and get the linear loop program $L^{\mathcal{V}}$ (this is the projection to $\mathcal{V} + \boldsymbol{x}^{\mathcal{U}}$). Importantly, the projection of $\boldsymbol{z_0}, \boldsymbol{z_1}, \ldots$ to $\mathcal{V} + \boldsymbol{x}^{\mathcal{U}}$ is still an infinite execution, hence the loop $L^{\mathcal{V}}$ is nonterminating. Given a GNTA for $L^{\mathcal{V}}$ we can construct a GNTA for L by adding the vector $\boldsymbol{x}^{\mathcal{U}}$ to $\boldsymbol{x_0}$ and $\boldsymbol{x_1}$. □

Proof (of Theorem 11). The polyhedron corresponding to loop transition of the deterministic linear loop program L is

$$\begin{pmatrix} G & 0 \\ M & -I \\ -M & I \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x'} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{g} \\ -\boldsymbol{m} \\ \boldsymbol{m} \end{pmatrix}. \tag{6}$$

Define $\mathcal{Y}$ to be the convex cone spanned by the rays of the guard polyhedron:

$$\mathcal{Y} := \{\boldsymbol{y} \in \mathbb{R}^n \mid G\boldsymbol{y} \leq 0\}$$

Let $\overline{\mathcal{Y}}$ be the smallest linear subspace of $\mathbb{R}^n$ that contains $\mathcal{Y}$, i.e., $\overline{\mathcal{Y}} = \mathcal{Y} - \mathcal{Y}$ using pointwise subtraction, and let $\overline{\mathcal{Y}}^{\perp}$ be the linear subspace of $\mathbb{R}^n$ orthogonal to $\overline{\mathcal{Y}}$; hence $\mathbb{R}^n = \overline{\mathcal{Y}} \oplus \overline{\mathcal{Y}}^{\perp}$.

Let $P := \{\boldsymbol{x} \in \mathbb{R}^n \mid G\boldsymbol{x} \leq \boldsymbol{g}\}$ denote the guard polyhedron. Its projection $P^{\overline{\mathcal{Y}}^{\perp}}$ to the subspace $\overline{\mathcal{Y}}^{\perp}$ is again a polyhedron. By the decomposition theorem for polyhedra [33, Cor. 7.1b], $P^{\overline{\mathcal{Y}}^{\perp}} = Q + C$ for some polytope Q and some convex cone C. However, by definition of the subspace $\overline{\mathcal{Y}}^{\perp}$, the convex cone C must be equal to $\{\boldsymbol{0}\}$: for any $\boldsymbol{y} \in C \subseteq \overline{\mathcal{Y}}^{\perp}$, we have $G\boldsymbol{y} \leq \boldsymbol{0}$, thus $\boldsymbol{y} \in \mathcal{Y}$, and therefore $\boldsymbol{y}$ is orthogonal to itself, i.e., $\boldsymbol{y} = \boldsymbol{0}$. We conclude that $P^{\overline{\mathcal{Y}}^{\perp}}$ must be a polytope, and thus it is bounded. By assumption L is nonterminating, so $L^{\overline{\mathcal{Y}}^{\perp}}$ is nonterminating, and since $P^{\overline{\mathcal{Y}}^{\perp}}$ is bounded, any infinite execution of $L^{\overline{\mathcal{Y}}^{\perp}}$ must be bounded.

Let $\mathcal{U}$ denote the direct sum of the generalized eigenspaces for the eigenvalues $0 \leq \lambda < 1$. Any infinite execution is necessarily bounded on the subspace $\mathcal{U}$ since on this space the map $\boldsymbol{x} \mapsto M\boldsymbol{x}+\boldsymbol{m}$ is a contraction. Let $\mathcal{U}^{\perp}$ denote the subspace of $\mathbb{R}^n$ orthogonal to $\mathcal{U}$. The space $\overline{\mathcal{Y}} \cap \mathcal{U}^{\perp}$ is a linear subspace of $\mathbb{R}^n$ and any infinite execution in its complement is bounded. Hence we can turn our analysis to the subspace $\overline{\mathcal{Y}} \cap \mathcal{U}^{\perp} + \boldsymbol{x}$ for some $\boldsymbol{x} \in \overline{\mathcal{Y}}^{\perp} \oplus \mathcal{U}$ for the rest of the proof according to Lemma 12. From now on, we implicitly assume that we are in this space without changing any of the notation.

Part 1. In this part we show that there is a basis $\boldsymbol{y_1}, \ldots, \boldsymbol{y_k} \in \mathcal{Y}$ such that M turns into a matrix U of the form given in (1) with $\lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_{k-1} \geq 0$. Since we allow μ_i to be positive between different eigenvalues (Example 14 illustrates why), this is not necessarily a Jordan normal form and the vectors $\boldsymbol{y_i}$ are not necessarily generalized eigenvectors.

We choose a basis $\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}$ such that M is in Jordan normal form with the eigenvalues ordered by size such that the largest eigenvalues come first. Define $\mathcal{V}_1 := \overline{\mathcal{Y}} \cap \mathcal{U}^{\perp}$ and let $\mathcal{V}_1 \supset \ldots \supset \mathcal{V}_k$ be a strictly descending chain of linear subspaces where $\mathcal{V}_i$ is spanned by $\boldsymbol{v_i}, \ldots, \boldsymbol{v_k}$.

We define a basis $\boldsymbol{w_1}, \ldots, \boldsymbol{w_k}$ by doing the following for each Jordan block of M, starting with $i = 1$. Let $M^{(i)}$ be the projection of M to the linear subspace $\mathcal{V}_i$ and let λ be the largest eigenvalues of $M^{(i)}$. The m-fold iteration of a Jordan

block $J_\ell(\lambda)$ for $m \geq \ell$ is given by

$$J_\ell(\lambda)^m = \begin{pmatrix} \lambda^m & \binom{m}{1}\lambda^{m-1} & \dots & \binom{m}{\ell}\lambda^{m-\ell} \\ & \lambda^m & \dots & \binom{m}{\ell-1}\lambda^{m-\ell+1} \\ & & \ddots & \vdots \\ 0 & & & \lambda^m \end{pmatrix} \in \mathbb{R}^{\ell\times\ell}. \tag{7}$$

Let $\boldsymbol{z_0}, \boldsymbol{z_1}, \boldsymbol{z_2}, \dots$ be an infinite execution of the loop L in the basis $\boldsymbol{v_i}, \dots, \boldsymbol{v_k}$ projected to the space $\mathcal{V}_i$. Since by Lemma 12 we can assume that there are no fixed points on this space, $|\boldsymbol{z_t}| \to \infty$ as $t \to \infty$ in each of the top ℓ components. Asymptotically, the largest eigenvalue λ dominates and in each row of $J_k(\lambda_i)^m$ (7), the entries $\binom{m}{j}\lambda^{m-j}$ in the rightmost column grow the fastest with an asymptotic rate of $\Theta(m^j \exp(m))$. Therefore the sign of the component corresponding to basis vector $\boldsymbol{v_{i+\ell}}$ determines whether the top ℓ entries tend to $+\infty$ or $-\infty$, but the top ℓ entries of $\boldsymbol{z_t}$ corresponding to the top Jordan block will all have the same sign eventually. Because no state can violate the guard condition we have that the guard cannot constraint the infinite execution in the direction of $\boldsymbol{v_j}$ or $-\boldsymbol{v_j}$, i.e., $G^{\mathcal{V}_i}\boldsymbol{v_j} \leq \boldsymbol{0}$ for each $i \leq j \leq i+\ell$ or $G^{\mathcal{V}_i}\boldsymbol{v_j} \geq \boldsymbol{0}$ for each $i \leq j \leq i+\ell$, where $G^{\mathcal{V}_i}$ is the projection of G to the subspace $\mathcal{V}_i$. So without loss of generality the former holds (otherwise we use $-\boldsymbol{v_j}$ instead of $\boldsymbol{v_j}$ for $i \leq j \leq i+\ell$) and for $i \leq j \leq i+\ell$ we get $\boldsymbol{v_j} \in \mathcal{Y} + \mathcal{V}_i^\perp$ where $\mathcal{V}_i^\perp$ is the space spanned by $\boldsymbol{v_1}, \dots, \boldsymbol{v_{i-1}}$. Hence there is a $\boldsymbol{u_j} \in \mathcal{V}_i^\perp$ such that $\boldsymbol{w_j} := \boldsymbol{v_j} + \boldsymbol{u_j}$ is an element of $\mathcal{Y}$. Now we move on to the subspace $\mathcal{V}_{i+\ell+1}$, discarding the top Jordan block.

Let T be the matrix M written in the basis $\boldsymbol{w_1}, \dots, \boldsymbol{w_k}$. Then T is of upper triangular form: whenever we apply $M\boldsymbol{w_i}$ we get $\lambda_i\boldsymbol{w_i} + \boldsymbol{u_i}$ ($\boldsymbol{w_i}$ was an eigenvector in the space $\mathcal{V}_i$) where $\boldsymbol{u_i} \in \mathcal{V}_i^\perp$, the space spanned by $\boldsymbol{v_1}, \dots, \boldsymbol{v_{i-1}}$ (which is identical with the space spanned by $\boldsymbol{w_1}, \dots, \boldsymbol{w_{i-1}}$). Moreover, since we processed every Jordan block entirely, we have that for $\boldsymbol{w_i}$ and $\boldsymbol{w_j}$ from the same generalized eigenspace ($T_{i,i} = T_{j,j}$) that for $i > j$

$$T_{j,i} \in \{0,1\} \text{ and } T_{j,i} = 1 \text{ implies } i = j+1. \tag{8}$$

In other words, when projected to any generalized eigenspace T consists only of Jordan blocks.

Now we change basis again in order to get the upper triangular matrix U defined in (1) from T. For this we define the vectors

$$\boldsymbol{y_i} := \boldsymbol{\beta}_i \sum_{j=1}^{i} \alpha_{i,j}\boldsymbol{w_j}.$$

with nonnegative real numbers $\alpha_{i,j} \geq 0$, $\alpha_{i,i} > 0$, and $\boldsymbol{\beta} > 0$ to be determined later. Define the matrices $W := (\boldsymbol{w_1} \dots \boldsymbol{w_k})$, $Y := (\boldsymbol{y_1} \dots \boldsymbol{y_k})$, and $\alpha := (\alpha_{i,j})_{1\leq j\leq i\leq k}$. So α is a nonnegative lower triangular matrix with a positive diagonal and hence invertible. Since α and W are invertible, the matrix $Y = \operatorname{diag}(\boldsymbol{\beta})\alpha W$ is invertible as well and thus the vectors $\boldsymbol{y_1}, \dots, \boldsymbol{y_k}$ form a

basis. Moreover, we have $\boldsymbol{y_i} \in \mathcal{Y}$ for each i since $\alpha \geq 0$, $\boldsymbol{\beta} > 0$, and $\mathcal{Y}$ is a convex cone. Therefore we get

$$GY \leq 0. \tag{9}$$

We will first choose α. Define $T =: D + N$ where $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$ is a diagonal matrix and N is nilpotent. Since $\boldsymbol{w_1}$ is an eigenvector of M we have $M\boldsymbol{y_1} = M\boldsymbol{\beta}_1\alpha_{1,1}\boldsymbol{w_1} = \lambda_1\boldsymbol{\beta}_1\alpha_{1,1}\boldsymbol{w_1} = \lambda_1\boldsymbol{y_1}$. To get the form in (1), we need for all $i > 1$

$$M\boldsymbol{y_i} = \lambda_i\boldsymbol{y_i} + \mu_{i-1}\boldsymbol{y_{i-1}}. \tag{10}$$

Written in the basis $\boldsymbol{w_1}, \ldots, \boldsymbol{w_k}$ (i.e., multiplied with W^{-1}),

$$(D+N)\boldsymbol{\beta}_i \sum_{j \leq i} \alpha_{i,j}\boldsymbol{e_j} = \lambda_i\boldsymbol{\beta}_i \sum_{j \leq i} \alpha_{i,j}\boldsymbol{e_j} + \mu_{i-1}\boldsymbol{\beta}_{i-1} \sum_{j<i} \alpha_{i-1,j}\boldsymbol{e_j}.$$

Hence we want to pick α such that

$$\sum_{j \leq i} \alpha_{i,j}(\lambda_j - \lambda_i)\boldsymbol{e_j} + N \sum_{j \leq i} \alpha_{i,j}\boldsymbol{e_j} - \mu_{i-1}\boldsymbol{\beta}_{i-1} \sum_{j<i} \alpha_{i-1,j}\boldsymbol{e_j} = \boldsymbol{0}. \tag{11}$$

First note that these constraints are independent of $\boldsymbol{\beta}$ if we set $\mu_{i-1} := \boldsymbol{\beta}_{i-1}^{-1} > 0$, so we can leave assigning a value to $\boldsymbol{\beta}$ to a later part of the proof.

We distinguish two cases. First, if $\lambda_{i-1} \neq \lambda_i$, then $\lambda_j - \lambda_i$ is positive for all $j < i$ because larger eigenvalues come first. Since N is nilpotent and upper triangular, $N \sum_{j \leq i} \alpha_{i,j}\boldsymbol{e_j}$ is a linear combination of $\boldsymbol{e_1}, \ldots, \boldsymbol{e_{i-1}}$ (i.e., only the first $i-1$ entries are nonzero). Whatever values this vector assumes, we can increase the parameters $\alpha_{i,j}$ for $j < i$ to make (11) larger and increase the parameters $\alpha_{i-1,j}$ for $j < i$ to make (11) smaller.

Second, let ℓ be minimal such that $\lambda_\ell = \lambda_i$ with $\ell \neq i$, then $\boldsymbol{w_\ell}, \ldots, \boldsymbol{w_j}$ are from the same generalized eigenspace. For the rows $1, \ldots, \ell - 1$ we can proceed as we did in the first case and for the rows $\ell, \ldots, i-1$ we note that by (8) $N\boldsymbol{e_j} = T_{j-1,j}\boldsymbol{e_{j-1}}$. Hence the remaining constraints (11) are

$$\sum_{\ell < j \leq i} \alpha_{i,j}T_{j-1,j}\boldsymbol{e_{j-1}} - \mu_{i-1} \sum_{\ell \leq j < i} \alpha_{i-1,j}\boldsymbol{e_j} = \boldsymbol{0},$$

which is solved by $\alpha_{i,j+1}T_{j,j+1} = \alpha_{i-1,j}$ for $\ell \leq j < i$. This is only a problem if there is a j such that $T_{j-1,j} = 0$, i.e., if there are multiple Jordan blocks for the same eigenvalue. In this case, we can reduce the dimension of the generalized eigenspace to the dimension of the largest Jordan block by combining all Jordan blocks: if $M\boldsymbol{y_i} = \lambda\boldsymbol{y_i} + \boldsymbol{y_{i-1}}$, and $M\boldsymbol{y_j} = \lambda\boldsymbol{y_j} + \boldsymbol{y_{j-1}}$, then $M(\boldsymbol{y_i} + \boldsymbol{y_j}) = \lambda(\boldsymbol{y_i} + \boldsymbol{y_j}) + (\boldsymbol{y_{i-1}} + \boldsymbol{y_{j-1}})$ and if $M\boldsymbol{y_i} = \lambda\boldsymbol{y_i} + \boldsymbol{y_{i-1}}$, and $M\boldsymbol{y_j} = \lambda\boldsymbol{y_j}$, then $M(\boldsymbol{y_i} + \boldsymbol{y_j}) = \lambda(\boldsymbol{y_i} + \boldsymbol{y_j}) + \boldsymbol{y_{i-1}}$. In both cases we can replace the basis vector $\boldsymbol{y_i}$ with $\boldsymbol{y_i} + \boldsymbol{y_j}$ without reducing the expressiveness of the GNTA.

Importantly, there are no cyclic dependencies in the values of α because neither one of the coefficients α can be made too large. Therefore we can choose $\alpha \geq 0$ such that (10) is satisfied for all $i > 1$ and hence the basis $\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}$ brings M into the desired form (1).

Part 2. In this part we construct the geometric nontermination argument and check the constraints from Definition 5. Since L has an infinite execution, there is a point $\boldsymbol{x}$ that fulfills the guard, i.e., $G\boldsymbol{x} \leq \boldsymbol{g}$. We choose $\boldsymbol{x_1} := \boldsymbol{x} + Y\boldsymbol{\gamma}$ with $\boldsymbol{\gamma} \geq \boldsymbol{0}$ to be determined later. Moreover, we choose $\lambda_1, \ldots, \lambda_k$ and $\mu_1, \ldots, \mu_{k-1}$ from the entries of U given in (1). The size of our GNTA is k, the number of vectors $\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}$. These vectors form a basis of $\overline{\mathcal{Y}} \cap \mathcal{U}^\perp$, which is a subspace of $\mathbb{R}^n$; thus $k \leq n$, as required.

The constraint (domain) is satisfied by construction and the constraint (init) is vacuous since L is a loop program. For (ray) note that from (9) and (10) we get

$$\begin{pmatrix} G & 0 \\ M & -I \\ -M & I \end{pmatrix} \begin{pmatrix} \boldsymbol{y_i} \\ \lambda_i \boldsymbol{y_i} + \mu_{i-1} \boldsymbol{y_{i-1}} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}.$$

The remainder of this proof shows that we can choose $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ such that (point) is satisfied, i.e., that

$$G\boldsymbol{x_1} \leq \boldsymbol{g} \text{ and } M\boldsymbol{x_1} + \boldsymbol{m} = \boldsymbol{x_1} + Y\boldsymbol{1}. \tag{12}$$

The vector $\boldsymbol{x_1}$ satisfies the guard since $G\boldsymbol{x_1} = G\boldsymbol{x} + GY\boldsymbol{\gamma} \leq \boldsymbol{g} + \boldsymbol{0}$ according to (9), which yields the first part of (12). For the second part we observe the following.

$$\begin{aligned} & M\boldsymbol{x_1} + \boldsymbol{m} = \boldsymbol{x_1} + Y\boldsymbol{1} \\ \iff \quad & (M - I)(\boldsymbol{x} + Y\boldsymbol{\gamma}) + \boldsymbol{m} = Y\boldsymbol{1} \\ \iff \quad & (M - I)\boldsymbol{x} + \boldsymbol{m} = Y\boldsymbol{1} - (M - I)Y\boldsymbol{\gamma} \end{aligned}$$

Since Y is a basis, it is invertible, so

$$\begin{aligned} \iff \quad & Y^{-1}(M - I)\boldsymbol{x} + Y^{-1}\boldsymbol{m} = \boldsymbol{1} - Y^{-1}(M - I)Y\boldsymbol{\gamma} \\ \iff \quad & (U - I)Y^{-1}\boldsymbol{x} + Y^{-1}\boldsymbol{m} = \boldsymbol{1} - (U - I)\boldsymbol{\gamma} \\ \iff \quad & (U - I)\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{m}} = \boldsymbol{1} - (U - I)\boldsymbol{\gamma} \end{aligned} \tag{13}$$

with $\tilde{\boldsymbol{x}} := Y^{-1}\boldsymbol{x} = W^{-1}\alpha^{-1}\mathrm{diag}(\boldsymbol{\beta})^{-1}\boldsymbol{x}$ and $\tilde{\boldsymbol{m}} := Y^{-1}\boldsymbol{m} = W^{-1}\alpha^{-1}\mathrm{diag}(\boldsymbol{\beta})^{-1}\boldsymbol{m}$. Equation (13) is now conveniently in the basis $\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}$ and all that remains to show is that we can choose $\boldsymbol{\gamma} \geq \boldsymbol{0}$ and $\boldsymbol{\beta} > 0$ such that (13) is satisfied.

We proceed for each (not quite Jordan) block of U separately, i.e., we assume that we are looking at the subspace $\boldsymbol{y_j}, \ldots, \boldsymbol{y_i}$ with $\mu_i = \mu_{j-1} = 0$ and $\mu_k > 0$ for all $j \leq k < i$. If this space only contains eigenvalues that are larger than 1, then $U - I$ is invertible and has only nonnegative entries. By using large enough values for $\boldsymbol{\beta}$, we can make $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{m}}$ small enough, such that $\boldsymbol{1} \geq (U - I)\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{m}}$. Then we just need to pick $\boldsymbol{\gamma}$ appropriately.

If there is at least one eigenvalue 1, then $U - I$ is not invertible, so (13) could be overconstraint. Notice that $\mu_\ell > 0$ for all $j \leq \ell < i$, so only the bottom entry in the vector equation (13) is not covered by $\boldsymbol{\gamma}$. Moreover, since eigenvalues are ordered in decreasing order and all eigenvalues in our current subspace are ≥ 1,

we conclude that the eigenvalue for the bottom entry is 1. (Furthermore, i is the highest index since each eigenvalue occurs only in one block). Thus we get the equation $\tilde{\boldsymbol{m}}_i = 1$. If $\tilde{\boldsymbol{m}}_i$ is positive, this equation has a solution since we can adjust $\boldsymbol{\beta}_i$ accordingly. If it is zero, then the execution on the space spanned by $\boldsymbol{y_i}$ is bounded, which we can rule out by Lemma 12.

It remains to rule out that $\tilde{\boldsymbol{m}}_i$ is negative. Let $\mathcal{U}$ be the generalized eigenspace to the eigenvector 1 and use Lemma 13 below to conclude that $\boldsymbol{o} := N^{k-1}\boldsymbol{m} + \boldsymbol{u} \in \mathcal{Y}$ for some $\boldsymbol{u} \in \mathcal{U}^{\perp}$. We have that $M\boldsymbol{o} = M(N^{k-1}\boldsymbol{m} + \boldsymbol{u}) = M\boldsymbol{u} \in \mathcal{U}^{\perp}$, so $\boldsymbol{o}$ is a candidate to pick for the vector $\boldsymbol{w_i}$. Therefore without loss of generality we did so in part 1 of this proof and since $\boldsymbol{y_i}$ is in the convex cone spanned by the basis $\boldsymbol{w_1}, \ldots, \boldsymbol{w_k}$ we get $\tilde{\boldsymbol{m}}_i > 0$. □

Lemma 13 (Deterministic Loops with Eigenvalue 1). *Let $M = I + N$ and let N be nilpotent with nilpotence index k ($k := \min\{i \mid N^i = 0\}$). If $GN^{k-1}\boldsymbol{m} \not\leq \boldsymbol{0}$, then L is terminating.*

Proof. We show termination by providing an k-nested ranking function [26, Def. 4.7]. By [26, Lem. 3.3] and [26, Thm. 4.10], this implies that L is terminating.

According to the premise, $GN^{k-1}\boldsymbol{m} \not\leq 0$, hence there is at least one positive entry in the vector $GN^{k-1}\boldsymbol{m}$. Let $\boldsymbol{h}$ be a row vector of G such that $\boldsymbol{h}^T N^{k-1}\boldsymbol{m} =: \delta > 0$, and let $h_0 \in \mathbb{R}$ be the corresponding entry in $\boldsymbol{g}$. Let $\boldsymbol{x}$ be any state and let $\boldsymbol{x'}$ be a next state after the loop transition, i.e., $\boldsymbol{x'} = M\boldsymbol{x} + \boldsymbol{m}$. Define the affine-linear functions $f_j(\boldsymbol{x}) := -\boldsymbol{h}^T N^{k-j}\boldsymbol{x} + c_j$ for $1 \leq j \leq k$ with constants $c_j \in \mathbb{R}$ to be determined later. Since every state $\boldsymbol{x}$ satisfies the guard we have $\boldsymbol{h}^T\boldsymbol{x} \leq h_0$, hence $f_k(\boldsymbol{x}) = -\boldsymbol{h}^T\boldsymbol{x} + c_k \geq -h_0 + c_k > 0$ for $c_k := h_0 + 1$.

$$\begin{aligned} f_1(\boldsymbol{x'}) = f_1(\boldsymbol{x} + N\boldsymbol{x} + \boldsymbol{m}) &= -\boldsymbol{h}^T N^{k-1}(\boldsymbol{x} + N\boldsymbol{x} + \boldsymbol{m}) + c_1 \\ &= f_1(\boldsymbol{x}) - \boldsymbol{h}^T N^k \boldsymbol{x} - \boldsymbol{h}^T N^{k-1}\boldsymbol{m} \\ &< f_1(\boldsymbol{x}) - 0 - \delta \end{aligned}$$

For $1 < j \leq k$,

$$\begin{aligned} f_j(\boldsymbol{x'}) = f_j(\boldsymbol{x} + N\boldsymbol{x} + \boldsymbol{m}) &= -\boldsymbol{h}^T N^{k-j}(\boldsymbol{x} + N\boldsymbol{x} + \boldsymbol{m}) + c_j \\ &= f_j(\boldsymbol{x}) + f_{j-1}(\boldsymbol{x}) - \boldsymbol{h}^T N^{k-j}\boldsymbol{m} - c_{j-1} \\ &< f_j(\boldsymbol{x}) + f_{j-1}(\boldsymbol{x}) \end{aligned}$$

for $c_{j-1} := -\boldsymbol{h}^T N^{k-j}\boldsymbol{m} - 1$. □

Example 14 (U is not in Jordan Form). The matrix U defined in (1) and used in the completeness proof is generally *not* the Jordan normal form of the loop's transition matrix M. Consider the following linear loop program.

$$\begin{array}{l} \textbf{while}\ \ (a - b \geq 0 \wedge b \geq 0\,):\\ \quad a\ :=\ 3a\,;\\ \quad b\ :=\ b + 1\,; \end{array}$$

This program is nonterminating because a grows exponentially and hence faster than b. It has the geometric nontermination argument

$$\boldsymbol{x_0} = \left(\begin{smallmatrix}9\\1\end{smallmatrix}\right), \quad \boldsymbol{x_1} = \left(\begin{smallmatrix}9\\1\end{smallmatrix}\right), \quad \boldsymbol{y_1} = \left(\begin{smallmatrix}12\\0\end{smallmatrix}\right), \quad \boldsymbol{y_2} = \left(\begin{smallmatrix}6\\1\end{smallmatrix}\right), \quad \lambda_1 = 3, \quad \lambda_2 = 1, \quad \mu_1 = 1.$$

The matrix corresponding to the linear loop update is

$$M = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}$$

which is diagonal (hence diagonalizable). Therefore M is already in Jordan normal form. The matrix U defined according to (1) is

$$U = \begin{pmatrix} 3 & 1 \\ 0 & 1 \end{pmatrix}.$$

The nilpotent component $\mu_1 = 1$ is important and there is no GTNA for this loop program where $\mu_1 = 0$ since the eigenspace to the eigenvector 1 is spanned by $(0\ 1)^T$ which is in $\overline{\mathcal{Y}}$, but not in $\mathcal{Y}$. $\Diamond$

5 Experiments

We implemented our method in a tool that is specialized for the analysis of lasso programs and called Ultimate LassoRanker. LassoRanker is used by Ultimate Büchi Automizer which analyzes termination of (general) C programs via the following approach [20]. Büchi Automizer iteratively picks lasso shaped paths in the control flow graph converts them to lasso programs and lets LassoRanker analyze them. In case LassoRanker was able to prove nontermination a real counterexample to termination was found, in case LassoRanker was able to provide a termination argument (e.g., a linear ranking function), Büchi Automizer continues the analysis, but only on lasso shaped paths for which the termination arguments obtained in former iterations are not applicable.

We applied Ultimate Büchi Automizer to the 631 benchmarks from the category Termination of the 5th software verification competition SV-COMP 2016 [4] in two different settings: one setting where we use our geometric nontermination arguments (GNTA) and one setting where we only synthesize fixed points (i.e., infinite executions where one state is always repeated).

In both settings the constraints were stated over the integers and we used the SMT solver Z3 [21] with a timeout of 12s to solve our constraints. The overall timeout for the termination analysis was 60s. Using the fixed point setting the tool was able to solve 441 benchmarks and the overall time for solving the (linear) constraints was 56 seconds. Using the GNTA setting the tool was able to solve 487 benchmarks and the overall time for solving the (nonlinear) constraints 2349 seconds. The GNTA setting was able to solve 47 benchmarks that could not be solved using the fixed point setting because there was a nonterminating execution which did not had a fixed point. The fixed point setting was able to solve 2 benchmarks that could not be solved using the GNTA setting because in the latter setting solving the linear constraints took too long.

6 Related Work

One line of related work is focused on decidability questions for deterministic lasso programs. Tiwari [34] considered linear loop programs over the reals where only strict inequalities are used in the guard and proved that termination is decidable. Braverman [5] generalized this result to loop programs that use strict and non-strict inequalities in the guard. Furthermore, he proved that termination is also decidable for homogeneous deterministic loop programs over the integers. Rebiha et al. [32] generalized the result to integer loops where the update matrix has only real eigenvalues. Ouaknine et al. [27] generalized the result to integer lassos where the update matrix of the loop is diagonalizable.

Another line of related work is also applicable to nondeterministic programs and uses a constraint-based synthesis of recurrence sets. The recurrence sets are defined by templates [35,18] or the constraint is given in a second order theory for bit vectors [15]. These approaches can be used to find nonterminating lassos that do not have a geometric nontermination argument; however, this comes at the price that for nondeterministic programs an $\exists\forall\exists$-constraint has to be solved.

Furthermore, there is a long line of research [7,2,9,11,24,15,25,8] that addresses programs that are more general than lasso programs.

7 Conclusion

We presented a new approach to nontermination analysis for linear lasso programs. This approach is based on geometric nontermination arguments, which are an explicit representation of an infinite execution. These nontermination arguments can be found by solving a set of nonlinear constraints. In Section 4 we showed that the class of nonterminating linear lasso programs that have a geometric nontermination argument is quite large: it contains at least every deterministic linear loop program whose eigenvalues are nonnegative. We expect that this statement can be extended to encompass also negative and complex eigenvalues.

The synthesis of nontermination arguments is useful not only to discover infinite loops in program code, but also to accelerate termination analysis (a nonterminating lasso does not need to be checked for a termination argument) or overflow analysis. Furthermore, since geometric nontermination arguments readily provide an infinite execution, a discovered software fault is transparent to the user.

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. M. F. Atig, A. Bouajjani, M. Emmi, and A. Lal. Detecting fair non-termination in multithreaded programs. In P. Madhusudan and S. A. Seshia, editors, *CAV 2012*, volume 7358 of *Lecture Notes in Computer Science*, pages 210–226. Springer, 2012.

3. A. M. Ben-Amram and S. Genaim. Ranking functions for linear-constraint loops. In *POPL*, 2013.
4. D. Beyer. Reliable and reproducible competition results with benchexec and witnesses (report on sv-comp 2016). In *TACAS*, Lecture Notes in Computer Science. Springer, 2016.
5. M. Braverman. Termination of integer linear programs. In *CAV*, pages 372–385. Springer, 2006.
6. M. Brockschmidt, B. Cook, and C. Fuhs. Better termination proving through cooperation. In *CAV*, pages 413–429. Springer, 2013.
7. M. Brockschmidt, T. Ströder, C. Otto, and J. Giesl. Automated detection of non-termination and nullpointerexceptions for java bytecode. In *FoVeOOS*, volume 7421 of *Lecture Notes in Computer Science*, pages 123–141. Springer, 2011.
8. T. C. Urban, A. Gurfinkel. Synthesizing ranking functions from bits and pieces. In *TACAS*, Lecture Notes in Computer Science. Springer, 2016.
9. H.-Y. Chen, B. Cook, C. Fuhs, K. Nimkar, and P. O'Hearn. Proving nontermination via safety. In *TACAS*, 2014.
10. B. Cook, J. Fisher, E. Krepska, and N. Piterman. Proving stabilization of biological systems. In *VMCAI*, pages 134–149, 2011.
11. B. Cook, C. Fuhs, K. Nimkar, and P. W. O'Hearn. Disproving termination with overapproximation. In *FMCAD 2014*, pages 67–74. IEEE, 2014.
12. B. Cook, H. Khlaaf, and N. Piterman. On automation of ctl* verification for infinite-state systems. In *CAV 2015*, pages 13–29, 2015.
13. B. Cook, E. Koskinen, and M. Y. Vardi. Temporal property verification as a program analysis task. In *CAV 2011*, pages 333–348, 2011.
14. B. Cook, A. Podelski, and A. Rybalchenko. Terminator: Beyond safety. In *CAV*, pages 415–418, 2006.
15. C. David, D. Kroening, and M. Lewis. Unrestricted termination and non-termination arguments for bit-vector programs. In *ESOP*, volume 9032 of *Lecture Notes in Computer Science*, pages 183–204. Springer, 2015.
16. D. Dietsch, M. Heizmann, V. Langenfeld, and A. Podelski. Fairness modulo theory: A new approach to LTL software model checking. In *CAV 2015*, pages 49–66, 2015.
17. S. Gulwani and F. Zuleger. The reachability-bound problem. In *PLDI*, pages 292–304, 2010.
18. A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, and R.-G. Xu. Proving non-termination. In *POPL*, pages 147–158, 2008.
19. W. R. Harris, A. Lal, A. V. Nori, and S. K. Rajamani. Alternation for termination. In *SAS*, pages 304–319, 2010.
20. M. Heizmann, J. Hoenicke, and A. Podelski. Termination analysis by learning terminating programs. In *CAV*, pages 797–813, 2014.
21. D. Jovanovic and L. M. de Moura. Solving non-linear arithmetic. In *IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.
22. D. Kroening, N. Sharygina, S. Tonetta, A. Tsitovich, and C. M. Wintersteiger. Loop summarization using abstract transformers. In *ATVA*, pages 111–125, 2008.
23. D. Kroening, N. Sharygina, A. Tsitovich, and C. M. Wintersteiger. Termination analysis with compositional transition invariants. In *CAV*, pages 89–103, 2010.
24. D. Larraz, K. Nimkar, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Proving non-termination using max-smt. In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 779–796. Springer, 2014.
25. T. C. Le, S. Qin, and W. Chin. Termination and non-termination specification inference. In *PLDI*, pages 489–498. ACM, 2015.

26. J. Leike and M. Heizmann. Ranking templates for linear loops. *Logical Methods in Computer Science*, 11(1):1–27, 2015.
27. J. Ouaknine, J. S. Pinto, and J. Worrell. On termination of integer linear loops. In *Symposium on Discrete Algorithms*, pages 957–969, 2015.
28. A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, pages 239–251. Springer, 2004.
29. A. Podelski and A. Rybalchenko. Transition invariants. In *LICS*, pages 32–41, 2004.
30. A. Podelski and A. Rybalchenko. Transition predicate abstraction and fair termination. In *POPL*, pages 132–144, 2005.
31. A. Podelski and S. Wagner. A sound and complete proof rule for region stability of hybrid systems. In *HSCC*, pages 750–753, 2007.
32. R. Rebiha, N. Matringe, and A. V. Moura. Characterization of termination for linear homogeneous programs. Technical report, Institute of Computing, University of Campinas, March 2014.
33. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1999.
34. A. Tiwari. Termination of linear programs. In *CAV*, pages 70–82. Springer, 2004.
35. H. Velroyen and P. Rümmer. Non-termination checking for imperative programs. In *TAP*, volume 4966 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 2008.

Appendix

A List of Notation

$\mathbb{R}$	the set of real numbers
L	a linear lasso program
$\mathbf{0}$	a vector of zeros (of the appropriate dimension)
$\mathbf{1}$	a vector of ones (of the appropriate dimension)
$\boldsymbol{e_i}$	the i-th unit vector
n	number of variables in the loop program
k, i, j, ℓ	natural numbers
α, β, γ	various nonnegative/positive parameters
G	part of the program guard $G\boldsymbol{x} \leq \boldsymbol{g}$
M	the linear map in a deterministic linear loop program
$\boldsymbol{x}$	a program state, i.e., a real-valued vector of dimension n
$\boldsymbol{x}^*$	a fixed point of the loop transition
N	a nilpotent matrix
D	a diagonal matrix
T, U	upper triangular matrixes
λ_i	an eigenvalue
$\boldsymbol{y}$	a ray of the guard polyhedron, $G\boldsymbol{y} \leq 0$