Termination of Linear Programs

Ashish Tiwari*

SRI International,
333 Ravenswood Ave,
Menlo Park, CA, U.S.A
tiwari@csl.sri.com

Abstract. We show that termination of a class of linear loop programs is decidable. Linear loop programs are discrete-time linear systems with a loop condition governing termination, that is, a while loop with linear assignments. We relate the termination of such a simple loop, on all initial values, to the eigenvectors corresponding to only the positive real eigenvalues of the matrix defining the loop assignments. This characterization of termination is reminiscent of the famous stability theorems in control theory that characterize stability in terms of eigenvalues.

1 Introduction

Dynamical systems have been studied by both computer scientists and control theorists, but both the models and the properties studied have been different. However there is one class of models, called "discrete-time linear systems" in the control world, where there is a considerable overlap. In computer science, these are unconditional `while` loops with linear assignments to a set of integer or rational variables; for example,

`while` $(true)$ $\{\ x := x - y; y := y\ \}$.

The two communities are interested in different questions: stability and controllability issues in control theory against reachability, invariants, and termination issues in computer science. In recent years, computer scientists have begun to apply the rich mathematical knowledge that has been developed in systems theory for analyzing such systems for safety properties, see for instance [17, 12, 11].

One of the most basic results in the theory of linear systems, both discrete-time and continuous-time, is the characterization of the stability of linear systems in terms of the eigenvalues of the corresponding matrix. In this paper, we are interested in termination of simple `while` loop programs, such as the one described above, but with nontrivial loop guards. We present results that relate the termination of such linear programs to eigenvalues of the corresponding matrix, analogous to the stability characterization in control theory. Our characterization also yields decidability of the termination problem for such programs. Although

* Research supported in part by the National Science Foundation under grant CCR-0311348 and CCR-0326540.

linear programs are similar to discrete-time linear systems, the termination characterization of linear programs is more complex than, though reminiscent of, the stability characterization for both continuous- and discrete-time linear systems.

Linear loop programs, as studied in this paper, are specialized piecewise affine systems, which themselves are special kinds of nonlinear systems. While several properties, such as reachability, stability, and controllability, are decidable for linear systems [10, 16], they soon become undecidable even when a "little" nonlinearity is introduced [16]. In particular, this is also true for piecewise affine systems [3, 4], see also Section 6. In this context, it is interesting to note that termination is decidable for linear loop programs.

Techniques to prove termination of programs have attracted renewed attention lately [7, 5, 6, 13]. The popular approach to prove termination is through the synthesis of a ranking function, a mapping from the state space to a well-founded domain, whose value monotonically decreases as the system moves forward. This line of research has focused mostly on generating *linear* ranking functions – some effective heuristics have been proposed [5, 6] and recently a complete method was presented in [13] for a model motivated by [14]. This paper investigates termination at a more basic theoretical level. The main result establishes the decidability of the termination problem for programs of the form (in matrix notation)

`while` $(B\boldsymbol{x} > \boldsymbol{b})\ \{\ \boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{c}\ \}$

where $B\boldsymbol{x} > \boldsymbol{b}$ represents a conjunction of linear inequalities over the state variables $\boldsymbol{x}$ and $\boldsymbol{x} := A\boldsymbol{x}+\boldsymbol{c}$ represents the linear assignments to each of the variables. The variables are interpreted over the reals $\Re$ and hence the state space is $\Re^n$. This class of programs is simpler than the ones considered in [5, 6, 13]. Although a program may not be presented in this form, termination questions can often be reduced to this basic form after suitable simplifications and transformations.

We approach the termination issue of the above program as follows. We first consider the homogeneous version,

`while` $(B\boldsymbol{x} > \boldsymbol{0})\ \{\ \boldsymbol{x} := A\boldsymbol{x}\ \}$,

and note that the condition $B\boldsymbol{x} > 0$ defines a region smaller than a half space of $\Re^n$. Now if a state $\boldsymbol{x} = \boldsymbol{c}$ is mapped by A (in one or more iterations) to something on the other side of the half space, then the program will terminate on this state (since the loop condition will become false). In particular, this means that the program always terminates on states specified by eigenvectors $\boldsymbol{c}$ corresponding to negative real eigenvalue and complex eigenvalues. The former ones are mapped by A to their negative image, while the latter ones are rotated gradually until they reach the other half space (where $B\boldsymbol{x} > 0$ is false). Thus, our first result is that, for purposes of termination, the eigenspace corresponding to only the *positive real* eigenvalues of A is relevant (Section 2 and Section 3).

In the case when all eigenvalues are positive, the eigenvectors corresponding to larger eigenvalues dominate the behavior of the program, that is, after sufficiently many iterations, the values of the state variables will be governed almost solely by the influence of the largest eigenvalue. Based on this, we can guess a witness to nontermination and test if the guess is correct by checking satisfiability of a set of constraints (Section 4). Finally, we show that the nonhomogeneous case can be reduced to the homogeneous case (Section 5).

1.1 Notation

We use standard mathematical notation for representing vectors and matrices. We follow the convention that upper case letters $I, J, \ldots$, denote integer constants and lower case letters $i, j, \ldots$ denote indices ranging over integers. In particular, a $(N \times 1)$ column matrix is called a vector, and it is denoted by $\boldsymbol{c}, \boldsymbol{d}$ whenever the components of the vector are known constants; and by $\boldsymbol{x}, \boldsymbol{y}$ whenever the components of the vector are all variables. A $(N \times N)$-matrix with constant entries a_{ij} at the (i, j)-position is denoted by $A = (a_{ij})$. A diagonal matrix $A = (a_{ij}) = diag(\lambda_1, \ldots, \lambda_N)$ has $a_{ii} = \lambda_i$ and $a_{ij} = 0$ otherwise. The transpose of a matrix $A = (a_{ij})$ is a matrix $B = (b_{ij})$ such that $b_{ij} = a_{ji}$, and it is denoted by A^T. Note that the transpose of a column vector $\boldsymbol{c}$ is a row vector $\boldsymbol{c}^T$. Using juxtaposition for matrix multiplication, we note that $\boldsymbol{c}^T\boldsymbol{d}$ denotes the inner product, $\sum_i c_i d_i$, of the vectors $\boldsymbol{c}$ and $\boldsymbol{d}$.

We will also denote matrices by specifying the submatrices inside it. So, for instance, $diag(J_1, \ldots, J_K)$ would denote a matrix which has matrices $J_1, \ldots, J_K$ on its "diagonal" and $\mathbf{0}$ elsewhere. If A is a $(N \times N)$-matrix and $\boldsymbol{c}$ is a vector such that $A\boldsymbol{c} = \lambda\boldsymbol{c}$, then $\boldsymbol{c}$ is called an *eigenvector* of A corresponding to the *eigenvalue* λ. The effect of repeated linear assignments $(\boldsymbol{x} := A\boldsymbol{x})$ becomes much more explicit when we do a change of variables and let the new variables $\boldsymbol{y}$ be the eigenvectors of A. In particular, we get transformed assignments of the form $y := \lambda y$. If there are N linearly independent eigenvectors, then A is said to be *diagonalizable* and the assignments on the new variables will be of the form $\boldsymbol{y} := diag(\lambda_1, \ldots, \lambda_N)\boldsymbol{y}$. But this is not possible always. However, instead of a diagonal matrix, we can always get an almost diagonal, the so-called *Jordan form*, matrix [9].

2 The Homogeneous Case

The presentation in this paper is incremental— going from syntactically simple to more complex programs. In this section, we consider linear programs of the following form:

`P1: while` $(\boldsymbol{c}^T\boldsymbol{x} > 0)$ $\{$ $\boldsymbol{x} := A\boldsymbol{x}$ $\}$.

The variables in $\boldsymbol{x}$ are interpreted over the set $\Re$ of reals. The assignment $\boldsymbol{x} := A\boldsymbol{x}$ is interpreted as being done simultaneously and not in any sequential order. A list of sequential assignments can be modified and presented in the form $\boldsymbol{x} := A\boldsymbol{x}$, see Example 1. We say that the Program P1 *terminates* if it terminates on *all* initial values in $\Re$ for the variables in $\boldsymbol{x}$.

Theorem 1. *If the linear loop program P1, defined by an $(N \times N)$-matrix A and a nonzero $N \times 1$-vector $\boldsymbol{c}$, is nonterminating then there exists a real eigenvector $\boldsymbol{v}$ of A, corresponding to positive eigenvalue, such that $\boldsymbol{c}^T\boldsymbol{v} \geq 0$.*

Proof. (Sketch) Suppose the linear loop program is nonterminating. Define the set NT of all points on which the program does not terminate.

$$NT = \{\boldsymbol{x} \in \Re^N : \boldsymbol{c}^T\boldsymbol{x} > 0, \boldsymbol{c}^T A\boldsymbol{x} > 0, \boldsymbol{c}^T A^2\boldsymbol{x} > 0, \ldots, \boldsymbol{c}^T A^i\boldsymbol{x} > 0, \ldots\}.$$

By assumption, $NT \neq \emptyset$. The set NT is also A-invariant, that is, if $\boldsymbol{v} \in NT$, then $A\boldsymbol{v} \in NT$. Note that NT is an affine subspace of $\Re^N$, that is, it is closed under addition and scalar multiplication by positive reals. Hence, it is convex[1]. Define $T = \Re^N - NT$ to be the set of all points where the program terminates. Define the boundary, ∂NT, of NT and T as the set of all $\boldsymbol{v}$ such that (for all ϵ) there exists a point in the ϵ-neighborhood of $\boldsymbol{v}$ that belongs to T and another that belongs to NT.

Let NT' be the completion of NT, that is, $NT' = NT \cup \partial NT$. Since NT is A-invariant, it means that A maps NT into NT. By continuity we have that A also maps NT' into NT'. Now, NT' is convex, and if we identify points $\boldsymbol{x}$ and $\boldsymbol{y}$, written as $\boldsymbol{x} \sim \boldsymbol{y}$, that are nonzero scalar multiples of each other ($\boldsymbol{x} = \lambda\boldsymbol{y}$), then the resulting set ($NT'/\sim$) is closed and bounded (as a subset of $\Re^{n-1}$). By Brouwer's fixed point theorem [15], it follows that there is an eigenvector $\boldsymbol{v}$ (with positive eigenvalue) of A in NT'. For all points $\boldsymbol{u} \in NT$, we know $\boldsymbol{c}^T\boldsymbol{u} > 0$. By continuity, for all points $\boldsymbol{u} \in NT'$, we have $\boldsymbol{c}^T\boldsymbol{u} \geq 0$. ∎

If, in fact, it is the case that $\boldsymbol{c}^T\boldsymbol{v} > 0$, then $\boldsymbol{v}$ is a witness to nontermination of the loop. Thus, Theorem 1 can be used to get the following conditional characterization of nontermination.

Corollary 1. *If there is no real eigenvector $\boldsymbol{v}$ of A such that $\mathbf{c}^T\boldsymbol{v} = 0$, then the linear loop program defined by A and $\mathbf{c}$ is nonterminating iff there exists an eigenvector $\boldsymbol{v}$ on which the loop is nonterminating.*

Example 1. The effect of two *sequential* assignments $x := x - y; y := x + 2y$ is captured by the *simultaneous* assignment

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

The matrix A has no real eigenvalues. Let $\boldsymbol{c} \neq \mathbf{0}$ be any (nonzero) vector. The condition of Corollary 1 is trivially satisfied. And since there is no real eigenvalue, we immediately conclude that the linear loop program specified by A and (any nonzero vector) $\boldsymbol{c}$ is terminating.

Example 2. Let θ be a fixed number. Consider the following program: `while` $(z - y > 0)$ { $\boldsymbol{x} := A\boldsymbol{x}$ }, where $A = [\cos\theta, -\sin\theta, 0; \sin\theta, \cos\theta, 0; 0, 0, 1]$. Thus, A simply rotates the 3-D space by an angle θ about the z-axis. The set of points where this program is nonterminating is $NT = \{(x, y, z) : z > x\sin\phi + y\cos\phi : \phi = n\theta, n = 0, 1, 2, \ldots\}$. For θ that is not a factor of π, $\partial NT = \{(x, y, z) : z^2 = x^2 + y^2\}$ (eliminate ϕ from above). Note that there is the eigenvector $(0, 0, 1)$ in NT corresponding to positive eigenvalue 1. As another example of the boundary, note that if $\theta = \pi/4$, then ∂NT contains 8 hyperplanes, each one is mapped by A to the next adjacent one.

[1] A set NT is convex if $\alpha\boldsymbol{u} + (1 - \alpha)\boldsymbol{v} \in NT$ whenever $\boldsymbol{u}, \boldsymbol{v} \in NT$ and $0 \leq \alpha \leq 1$.

2.1 Generalizing the Loop Condition

The loop condition can be generalized to allow for a conjunction of multiple linear inequalities. We continue to assume that all linear inequalities and linear assignments consist only of homogeneous expressions. Let B be a $(M \times N)$-matrix (with rational entries) and A be a $(N \times N)$-matrix. We consider programs of the following form:

`P2: while` $(B\boldsymbol{x} > 0)$ $\{\ \boldsymbol{x} := A\boldsymbol{x}\ \}$.

Theorem 1 and Corollary 1 immediately generalize to programs of the form P2.

Theorem 2. *If Program P2, specified by matrices A and B, is nonterminating, then there is a real eigenvector $\boldsymbol{v}$ of A, corresponding to a positive real eigenvalue, such that $B\boldsymbol{v} \geq 0$.*

Corollary 2. *Assume that for every real eigenvector $\boldsymbol{v}$ of A, corresponding to a positive eigenvalue, whenever $B\boldsymbol{v} \geq 0$, then it is actually the case that $B\boldsymbol{v} > 0$. Then, the Program P2, defined by A and B, is nonterminating iff there exists an eigenvector $\boldsymbol{v}$ on which the loop is nonterminating.*

Example 3. Consider the program:

`while` $(x - y > 0)$ $\{\ x := -x + y;\ y := y\ \}$.

The matrix $A = [-1, 1; 0, 1]$ has two eigenvalues, 1 and -1. The eigenvector corresponding to the eigenvalue 1 is $[1; 2]$ and we note that $1 - 2 \not> 0$. Hence, it follows from Corollary 2 that the above loop is terminating.

2.2 Two Variable Case

Theorem 1 and Theorem 2 show that nonterminating linear loops almost always have a witness that is an eigenvector of the matrix A. The only problematic case is when the eigenvector is on the boundary, ∂NT, so that it is not clear if indeed there are points where the program is nonterminating. However, in the 2-dimensional case, that is, when there are only two variables, the region NT will be a sector and it can be specified by its two boundary rays. Thus, if $NT \neq \emptyset$, then there exists an A-invariant sector, given by $\boldsymbol{a}^T\boldsymbol{x} \rhd 0 \ \wedge\ \boldsymbol{b}^T\boldsymbol{x} \rhd 0 \ \wedge\ \boldsymbol{x} \neq \boldsymbol{0}$ where $\rhd \in \{>, \geq\}$, on which the loop condition always evaluates to true. This can be expressed as a quantified formula over the theory of (linear) arithmetic interpreted over the reals, which is a decidable theory.

Theorem 3. *A two variable linear loop program,*

`while` $(B\boldsymbol{x} > 0)$ $\{\ \boldsymbol{x} := A\boldsymbol{x}\ \}$,

is non-terminating iff the following sentence in true in the theory of reals

$$\exists \boldsymbol{a}, \boldsymbol{b}.[\exists \boldsymbol{x}.\phi(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) \ \wedge\ \forall \boldsymbol{x}.(\phi(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) \ \Rightarrow\ (B\boldsymbol{x} > 0 \ \wedge\ \phi(\boldsymbol{a}, \boldsymbol{b}, A\boldsymbol{x})))]$$

where $\phi(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x})$ denotes $\boldsymbol{a}^T\boldsymbol{x} \rhd 0 \ \wedge\ \boldsymbol{b}^T\boldsymbol{x} \rhd 0 \ \wedge\ \boldsymbol{x} \neq \boldsymbol{0}$ and $\rhd \in \{>, \geq\}$.

This theorem gives a decision procedure for termination of two variable loops since the formula in Theorem 3 can be tested for satisfiability. Theorem 3 cannot be generalized to higher dimensions since there may not be finitely many hyperplane boundaries, as Example 2 illustrates.

3 Reducing the Homogeneous Case

Corollary 2 falls short of yielding decidability of termination of homogeneous linear programs. But it hints that the real eigenvalues and the corresponding eigenvectors are relevant for termination characteristics of such programs. In this section, we will formally show that the nonpositive eigenvalues (and the corresponding eigenspace) can be ignored and the termination problem can be reduced to only the eigenspace corresponding to positive real eigenvalues of the matrix A.

We first note that the Program P2 from Section 2.1 can be transformed by an invertible (bijective) transformation, preserving its termination properties.

Proposition 1. *Let P be an invertible linear transformation. The program*

P2: while $(B\boldsymbol{x} > 0)$ $\{$ $\boldsymbol{x} := A\boldsymbol{x}$ $\}$

is terminating iff the program

P3: while $(BP\boldsymbol{y} > 0)$ $\{$ $\boldsymbol{y} := P^{-1}AP\boldsymbol{y}$ $\}$

is terminating.

Proof. If Program P2 does not terminate on input $\boldsymbol{x} := \boldsymbol{c}$, then Program P3 will not terminate on input $\boldsymbol{y} := P^{-1}\boldsymbol{c}$. Conversely, if Program P3 does not terminate on input $\boldsymbol{y} := \boldsymbol{d}$, then Program P2 will not terminate on input $\boldsymbol{x} := P\boldsymbol{d}$. ∎

Thus, Proposition 1 is just about doing a "change of variables". It is a well known result in linear algebra [9, 1] that using a suitable change of variables, a real matrix A can be transformed into the form, $diag(J_1, J_2, \ldots, J_K)$, called the *real Jordan form*, where each J_i is either of the two forms:

$$\begin{pmatrix} \lambda_i & 1 & 0 \ldots & 0 \\ 0 & \lambda_i & 1 \ldots & 0 \\ & & & \\ 0 & 0 & \ddots & 1 \\ 0 & 0 & 0 \ldots & \lambda_i \end{pmatrix} \qquad \begin{pmatrix} D_i & I & 0 \ldots & 0 \\ 0 & D_i & I \ldots & 0 \\ & & & \\ 0 & 0 & \ddots & I \\ 0 & 0 & 0 \ldots & D_i \end{pmatrix}$$

where $\lambda_i \in \Re$ is a real whereas D_i is a (2×2)-matrix of the form $\begin{pmatrix} \alpha_i & -\beta_i \\ \beta_i & \alpha_i \end{pmatrix}$. For uniformity, the second Jordan block will denote both the forms. When it denotes the first form, then D_i and I are both (1×1)-matrices and we will say $D_i \in \Re$ and treat it as a real. We define $|D_i| = |\lambda_i|$ in the first case and $|D_i| = \sqrt{\alpha_i^2 + \beta_i^2}$ in the second case.

Let P be the real $(N \times N)$-matrix such that $P^{-1}AP = diag(J_1, \ldots, J_K)$. Thus, Program P2, specified by matrices A and B,

P2: while $(B\boldsymbol{x} > 0)$ $\{$ $\boldsymbol{x} := A\boldsymbol{x}$ $\}$,

can be transformed into the new Program P3,

P3: while $(BP\boldsymbol{y} > 0)$ $\{$ $\boldsymbol{y} := diag(J_1, \ldots, J_K)\boldsymbol{y}$ $\}$.

Proposition 1 means that we can focus on termination of Program P3. Partition the variables in $\boldsymbol{y}$ into $\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_K}$ and rewrite the Program P3 as

P3: while $(B_1\boldsymbol{y_1} + \cdots + B_K\boldsymbol{y_K} > 0)$ $\{$ $\boldsymbol{y_1} := J_1\boldsymbol{y_1}; \ldots; \boldsymbol{y_K} := J_K\boldsymbol{y_K}$ $\}$,

where B_i's are obtained by partitioning the matrix BP. Let $S = \{1, 2, \ldots, K\}$ be the set of indices. Define the set $S_+ = \{i \in S : D_i \in \Re, D_i > 0\}$. The following technical lemma shows that we can ignore the state space corresponding to negative and complex eigenvalues, while still preserving the termination behavior of the Program P3.

Lemma 1. *The Program P3, as defined above, is terminating iff the program*

`P4: while` $(\sum_{j \in S_+} B_j \boldsymbol{y_j} > 0)$ $\{$ $\boldsymbol{y_j} := J_j \boldsymbol{y_j}$; `for` $j \in S_+$ $\}$

is terminating.

Proof. (Sketch) If the Program P4 does not terminate on input $\boldsymbol{y_j} := \boldsymbol{c_j}$, where $j \in S_+$, then the Program P3 does not terminate on input $\boldsymbol{y_j} := \boldsymbol{c_j}$ for $j \in S_+$ and $\boldsymbol{y_j} := \mathbf{0}$ for $j \notin S_+$.

For the converse, assume that Program P3 does not terminate on input $\boldsymbol{y_j} := \boldsymbol{c_j}$, $j \in S$. Consider the m-th loop condition, $\sum_{j \in S} B_{jm} \boldsymbol{y_j} > 0$, where B_{jm} denotes the m-th row of B_j. Assume that $\boldsymbol{y_j}$ has N_j components, $\boldsymbol{y_{j0}}, \boldsymbol{y_{j1}}, \ldots, \boldsymbol{y_{jN_j-1}}$, where each $\boldsymbol{y_{jk}}$ is either a 2×1 or a 1×1 matrix (depending on whether D_j is 2×2 or 1×1.) The value of $\boldsymbol{y_j}$, at the i-th iteration, is given by

$$\boldsymbol{y_j}(i) = \begin{pmatrix} D_j^i & iD_j^{i-1} & \binom{i}{2} D_j^{i-2} & \ldots & \binom{i}{N_j - 1} D_j^{i-(N_j-1)} \\ 0 & D_j^i & iD_j^{i-1} & \ldots & \binom{i}{N_j - 2} D_j^{i-(N_j-2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & D_j^i & iD_j^{i-1} \\ 0 & 0 & \ldots & 0 & D_j^i \end{pmatrix} \boldsymbol{c_j} \qquad (1)$$

Define $f_m(i)$ to be the value of the expression of the m-th condition at i-th iteration, that is, $f_m(i) = \sum_{j \in S} B_{jm} \boldsymbol{y_j}(i)$. Let J be an index such that $N_J = max\{N_j : |D_j| = max\{|D_i| : i \in S\}\}$. We claim, without further proof, that for large i, the term $\binom{i}{N_J - 1} {D_J}^{i-N_{J'}+1} \boldsymbol{y_{J'N_{J'}-1}}(0)$ will dominate the value of $f_m(i)$. If $J \notin S_+$, then the sign of this dominating term, and consequently the sign of $f_m(i)$, will fluctuate (between positive and negative) as i increases. By assumption, this does not happen. Hence, $J \in S_+$ and hence, for a large enough i, say $i \geq I_m$, we have $\sum_{j \in S_+} B_{jm} \boldsymbol{y_j}(i) > 0$. For each condition m, we get an index I_m. Set I to be the maximum of all I_m's. For $i \geq I$, it is the case that $\sum_{j \in S_+} B_{jm} \boldsymbol{y_j}(i) > 0$ for all m.

Define new initial conditions for Program P3 and Program P4 as follows: $\boldsymbol{y_i}(0) := \boldsymbol{y_i}(I)$ for all $i \in S_+$ and $\boldsymbol{y_i}(0) := \mathbf{0}$ for all $i \notin S_+$. Program P3 does not terminate on this new initial conditions. Hence, Program P4 also does not terminate on it. This completes the proof. ∎

Example 4. We borrow the following example from [13],

`Q1: while` $(x > 0 \wedge y > 0)$ $\{$ $x := -2x + 10y$; $y := y$ $\}$.

The matrix A has two eigenvalues -2 and 1, and it is clearly diagonalizable. In fact, consider the transformation matrix P,

$$A = \begin{pmatrix} -2 & 10 \\ 0 & 1 \end{pmatrix} \qquad P = \begin{pmatrix} 1 & 10 \\ 0 & 3 \end{pmatrix} \qquad P^{-1}AP = \begin{pmatrix} -2 & 0 \\ 0 & 1 \end{pmatrix} \qquad BP = P$$

Transforming Program Q1 by P, we get

`Q2: while` $(x_1 + 10x_2 > 0 \ \wedge\ 3x_2 > 0)$ $\{\ x_1 := -2x_1;\ x_2 := x_2\ \}$.

Lemma 1 says that the termination of Program Q1 and Program Q2 can be decided by just considering the termination characteristics of:

`Q3: while` $(10x_2 > 0 \ \wedge\ 3x_2 > 0)$ $\{\ x_2 := x_2\ \}$.

In fact, the point $x_1 = 0, x_2 = 1$ makes Program Q3 nonterminating, and correspondingly, the point $x = 10, y = 3$ makes Program Q1 nonterminating.

4 All Positive Eigenvalues

Lemma 1 reduces the termination of Program P2 of Section 2.1 to testing termination of Program P4, which is given as:

`P4: while` $(B_1\boldsymbol{y_1} + \cdots + B_r\boldsymbol{y_r} > 0)$ $\{\ \boldsymbol{y_1} := J_1\boldsymbol{y_1}; \ldots ; \boldsymbol{y_r} := J_r\boldsymbol{y_r}\ \}$

where each of the Jordan blocks J_i corresponds to a *positive real* eigenvalue λ_i.

The value of variables in $\boldsymbol{y_j}$, after the i-th iteration, are given by Equation 1, where $D_j = \lambda_j$ is a positive real now. As before, assume that the k-th loop condition is written as $B_{1k}\boldsymbol{y_1} + B_{2k}\boldsymbol{y_2} + \cdots + B_{rk}\boldsymbol{y_r} > 0$. We can express the requirement that the k-th loop condition be true after the i-th iteration as

$$B_{1k}\boldsymbol{y_1}(i) + B_{2k}\boldsymbol{y_2}(i) + \cdots + B_{rk}\boldsymbol{y_r}(i) > 0.$$

Expand this using Equation 1 and let C_{klj} denote the result of collecting all coefficients of the term $\binom{i}{j-1}\lambda_l^{i-(j-1)}$. Now, the k-th loop condition after i-th iteration can be written as

$$\begin{aligned}\lambda_1^i C_{k11}\boldsymbol{y}(0) + i\lambda_1^{i-1}C_{k12}\boldsymbol{y}(0) + \cdots + \binom{i}{n_1 - 1}\lambda_1^{i-(n_1-1)}C_{k1n_1}\boldsymbol{y}(0) +&\\ \cdots +&\\ \lambda_r^i C_{kr1}\boldsymbol{y}(0) + i\lambda_r^{i-1}C_{kr2}\boldsymbol{y}(0) + \cdots + \binom{i}{n_r - 1}\lambda_r^{i-(n_r-1)}C_{krn_r}\boldsymbol{y}(0) > 0,&\end{aligned}$$

which we will denote by $Cond_k(\boldsymbol{y}(i))$. If two eigenvalues λ_l and λ_m are the same, then we assume that the corresponding coefficients (of $\binom{i}{j}\lambda_l^{i-j}$ and $\binom{i}{j}\lambda_m^{i-j}$) have been merged in the above expression, so that without loss of generality, we can assume that each λ_l is distinct and such that $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_r$.

Define the set $Ind = \{11, 12, \ldots, 1n_1, 21, 22, \ldots, 2n_2, \ldots, r1, r2, \ldots, rn_r\}$ and an ordering $\succ$ on this set so that elements on the right are greater-than elements on the left in the above set. The idea here is that nonzero terms in $Cond_k$ that have larger indices grow faster asymptotically (as i increases).

We decide the termination of Program P4 using the following three step nondeterministic algorithm:

(1) For each of the m loop conditions, we guess an element from the set Ind. Formally, we guess a mapping $index : \{1, 2, \ldots, m\} \mapsto Ind$. Intuitively if $\boldsymbol{y}(0)$ is a witness to nontermination, then $index(k)$ is chosen so that $C_{k,index(k)}\boldsymbol{y}(0) > 0$ (and this is the dominant summand) and $C_{k,ind}\boldsymbol{y}(0) = 0$ for all $ind \succ index(k)$.

(2) Build a set of linear equality and inequality constraints as follows: from the k-th loop condition, generate the following constraints,

$$\begin{array}{rl} C_{k,ind}\boldsymbol{z} = 0, & \text{if } ind \succ index(k) \\ C_{k,ind}\boldsymbol{z} > 0, & \text{if } ind = index(k) \\ Cond_k(\boldsymbol{z}(i)) > 0, & \text{if } 0 \leq i \leq \Pi_2(index(k)) \end{array}$$

where Π_1 and Π_2 denote the projection onto the first and second components respectively. Note that the unknowns are just $\boldsymbol{z}$ (the initial values for $\boldsymbol{y}$).

(3) Return "nonterminating" if the new set of linear inequalities and linear equations is satisfiable (in $\Re^n$), return "terminating" otherwise.

We state the correctness of the algorithm, without proof, in Lemma 2 and follow it up by a summary of the complete decision procedure in the proof of Theorem 4.

Lemma 2. *The nondeterministic procedure returns "nonterminating" iff the Program P4*

`P4: while` $(B_1\boldsymbol{y_1} + \cdots + B_r\boldsymbol{y_r} > 0)$ $\{\ \boldsymbol{y_1} := J_1\boldsymbol{y_1}; \ldots; \boldsymbol{y_r} := J_r\boldsymbol{y_r}\ \}$

is nonterminating.

Theorem 4. *The termination of a homogeneous linear program of the form*

`P2: while` $(B\boldsymbol{x} > 0)$ $\{\ \boldsymbol{x} := A\boldsymbol{x}\ \}$

is decidable.

Proof. We decide termination of the Program P2 as follows: If A has no positive real eigenvalues, then return "terminating" (Corollary 2). If every real eigenvector $\boldsymbol{v}$ corresponding to a positive real eigenvalue of A satisfies $B\boldsymbol{v} < 0$, then return "terminating" (Theorem 2). If there is a real eigenvector $\boldsymbol{v}$ corresponding to a positive real eigenvalue of A such that $B\boldsymbol{v} > 0$, then return "nonterminating" (Corollary 2). If none of the above cases is true, then clearly A has positive real eigenvalues. Compute the Jordan blocks and the generalized eigenvectors *only corresponding to the positive real eigenvalues* of A. Generate a transformation P by extending the computed set of generalized eigenvectors with *any* set of vectors in space orthogonal to that of the generated eigenvectors. Transform Program P2 by P as in Proposition 1. It is an easy exercise[2] to note that we can apply Lemma 1 and reduce the termination problem to that for Program P4 of Lemma 1. Finally, we decide termination of Program P4 using the nondeterministic procedure of Section 4 (Lemma 2). ■

Example 5. Consider the program

`Q4: while` $(x > 0 \ \wedge\ y > 0)$ `{ x := x - y; y := y }`

This contains only two variables, and hence we can use Theorem 3, but for purposes of illustration, we apply Theorem 4 here. The matrix $A = [1, -1; 0, 1]$ has a positive real eigenvalue 1. The vector given by $x = 1, y = 0$ is an eigenvector corresponding to this eigenvalue. Hence, we cannot apply Theorem 2 or

[2] We cannot apply Lemma 1 directly since we did not compute the real Jordan form of the "full" matrix A, but only of a part of it. But we do not need to compute the full real Jordan form to get Program P4.

Corollary 2. The real Jordan form of A is $A' = [1, 1; 0, 1]$ and the corresponding transformation matrix is $P = [-1, 0; 0, 1]$. The new program is:

Q5: while $(-x > 0 \ \wedge \ y > 0)$ { x := x + y; y := y }

The general solutions are given by

$$\begin{aligned} x(i) &= 1^i x(0) + iy(0) = x(0) + iy(0) \\ y(i) &= \qquad\quad 1^i y(0) = y(0) \end{aligned}$$

The condition $Cond_1$ corresponding to the loop condition $-x > 0$ is $-x(0) - iy(0) > 0$ and similarly $Cond_2$ is $y(0) > 0$. There is no choice for $index(2)$, but there is a choice for $index(1)$: it can be either 11 or 12.

In the first case, we generate the following constraints from the first loop condition: $\{-y = 0, -x > 0, -x - 0y > 0\}$. From the second loop condition, we only generate the constraint $y > 0$. Together, we detect an inconsistency.

In the second case, we generate the following constraints from the first loop condition: $\{-y > 0, -x - 0y > 0, -x - 1y > 0\}$. Again, from the second loop condition, we generate the constraint $y > 0$, which is inconsistent with the above constraints. Hence, we conclude that the Program Q4 is terminating.

5 Nonhomogeneous Programs

Now we consider linear programs of the following form:

P5: while $(B\boldsymbol{x} > \boldsymbol{b})$ { $\boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{c}$ }

We can homogenize this program and add an additional constraint on the homogenizing variable to get the following

P6: while $(B\boldsymbol{x} - \boldsymbol{b}z > 0 \ \wedge \ z > 0)$ { $\boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{c}z; z := z$ }

where z is a new variable. The homogeneous program reduces to the original program if we substitute 1 for z.

Proposition 2. *The nonhomogeneous Program P5 does not terminate iff the homogeneous Program P6 does not terminate.*

Proof. If Program P5 does not terminate, say on input $x_i = d_i, i = 1, 2, \ldots, n$, then Program P6 does not terminate on input $z = 1, x_i = d_i, i = 1, 2, \ldots, n$.

For the converse, assume that Program P6 does not terminate on input $x_0 = d_0, x_i = d_i, i = 1, 2, \ldots, n$. If $d_0 > 0$, then we can scale this input to get a new state $x_0 = 1, x_i = d_i/d_0, i = 1, 2, \ldots, n$. The behavior of Program P6 will be the same on the scaled input, and hence Program P5 would not terminate on this input either. ■

Thus we can reduce the decidability of termination of nonhomogeneous programs to that of homogeneous programs. Together with Theorem 4, we get the following result.

Theorem 5. *The termination of a nonhomogeneous linear program of the form*

P5: while $(B\boldsymbol{x} > \boldsymbol{b})$ { $\boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{c}$ }

is decidable.

Remarks on Computability and Complexity. The three step nondeterministic algorithm described in Section 4 is clearly in class NP. However, the nondeterminism can be eliminated by a careful enumeration of choices. The idea is that we always start with the guess $\mathit{index}(k) = rn_r$ for each loop condition k and if the resulting formula is unsatisfiable, then we readjust the guess and gradually set $\mathit{index}(k)$ to smaller elements in the set Ind. We do not formalize this detail in this paper because it is not central to the decidability result.

The reduction described in Section 3 requires computation of a real eigenvector. Computing with real numbers is not easy, but if the input matrices are over the rationals, then all the real numbers that arise in the computation are *algebraic*. Computing with algebraic numbers is theoretically possible, but it can be expensive (since the theory of real-closed fields has a double exponential lower bound). But this should not be a serious problem for two reasons. First, for most problems arising in practice, we expect the eigenvalues to be rational, and computation with rationals can be done very efficiently. Second, even if the eigenvalues are irrational, the dimension of the problem is unlikely to be so high that computing with algebraic numbers will become a bottleneck. However, these issues have to be experimented with and that is left as future work.

We believe that the Jordan form computation step in the decision procedure outlined in this paper can be eliminated in most cases in practice. This can be achieved by perturbing the system by a little (for example, replacing conditions $c > 0$ by $c > \epsilon$ for some small constant ϵ) and studying the termination property of the perturned system. We conjecture that Corollary 2 can be strengthened for the case when the homogenization variable violates the condition. This would allow us to avoid the Jordan decomposition step in many cases.

The decision procedure for termination of linear loop programs can be adapted to the case when the variables are interpreted over the integers and rationals. If there are "a lot" of witnesses to nontermination, then there will be a rational witness too, since the rationals are dense in reals. If not, then we can detect this case using a specialized wrapper. This leads us to conjecture the following.

Conjecture 1. The termination of Program P5, as defined in Theorem 5, when all variables are interpreted over the set of integers, is decidable.

6 The General Case

We consider the termination of a *set* of nondeterministic linear conditional assignments, written in a guarded command language [8] as,

$$\texttt{P7}: \quad \begin{array}{llll} [& B_1\boldsymbol{x} > \boldsymbol{b_1} & \longrightarrow & \boldsymbol{x} := A_1\boldsymbol{x} + \boldsymbol{c_1} \\ [] & B_2\boldsymbol{x} > \boldsymbol{b_2} & \longrightarrow & \boldsymbol{x} := A_2\boldsymbol{x} + \boldsymbol{c_2} \\ \ldots & & & \\ [] & B_k\boldsymbol{x} > \boldsymbol{b_k} & \longrightarrow & \boldsymbol{x} := A_k\boldsymbol{x} + \boldsymbol{c_k} \;] \end{array}$$

which we will write in shorthand as

$\texttt{P7:}\ \texttt{[]}_{i=1}^{k}(B_i\boldsymbol{x} > \boldsymbol{b_i} \longrightarrow \boldsymbol{x} := A_i\boldsymbol{x} + \boldsymbol{c_i})$

Counter machines can be naturally encoded as Program P7. We introduce one variable x_i for each counter and one variable x for the finite control (program counter). Encodings of conditional branches, counter increments, and counter decrements are straightforward. The problem of deciding if a counter machine halts on all inputs is undecidable, see [3], where this problem is called the mortality problem. Therefore, the problem of deciding if a program of the above form halts on all *integer* inputs is also undecidable. Note however that for the restricted forms of assignments ($x := x \pm 1$) generated by the translation, termination over reals is equivalent to termination over integers. Thus, we conclude that the *termination problem for Program P7 is undecidable.*

Theorem 5 can be used to get an incomplete test for nontermination for Program P7—If Program P7 is terminating, then for each i, the program `while` $(B_i\boldsymbol{x} > \boldsymbol{b_i})$ $\{\ \boldsymbol{x} := A_i\boldsymbol{x} + \boldsymbol{c_i}\ \}$ is terminating. The converse is also true under additional *commutation* properties [2] amongst the k binary relations, say $R_1, \ldots, R_k$, induced by the k guarded commands. In particular, one immediate consequence of a result in [2] is the following.

Proposition 3. *Let Program P7 and relations $R_1, \ldots, R_k$ be as defined above. Let $R = R_1 \cup \cdots \cup R_k$. Assume that whenever $i < j$, it is the case that $R_j \circ R_i \subseteq R_i \circ R^*$. Then, the Program P7 terminates if and only if each R_i do.*

Note that the condition above is dependent on the order $R_1, \ldots, R_k$, which we are free to choose. Testing for the quasi-commutation property [2] $R_j \circ R_i \subseteq R_i \circ R^*$ is possible if we restrict the search to $R_j \circ R_i \subseteq R_i \circ R^l$, for some finite l. In the special case when the i-th guarded command cannot be enabled after execution of the j-th guarded command (that is, $R_j \circ R_i = \emptyset$), then the above inclusion is trivially true.

In the case when the quasi-commutation property cannot be established, the test for nontermination can be made "more complete" by including new guarded transitions obtained by composing two or more of the original guarded commands. It is easy to see that the composition results in a *linear* guarded command. These new guarded commands can be tested for nontermination using Theorem 5 again.

7 Future Work and Conclusion

We have presented decidability results for termination of simple loop programs. The loops are considered terminating if they terminate on all initial real values of the variables. The decision procedure is based on the observation that only the eigenvectors (and the generalized eigenspace) corresponding to positive real eigenvalues of the assignment matrix are relevant for termination. The generalization to multiple linear nondeterministic guarded commands makes the problem undecidable. Under certain restrictive commutation conditions, termination of multiple linear guarded commands can be reduced to termination of each individual simple linear loops.

We believe that results and tools from systems theory, such as Lyapunov functions and control Lyapunov functions, can yield powerful tools for analyzing software, especially for termination analysis and invariant generation. This avenue should be explored further in the future.

Acknowledgments. We wish to thank Andreas Podelski, Andrey Rybalchenko, and their colleagues at MPI for motivation and initial discussions and the reviewers for insightful comments and references.

References

[1] D. K. Arrowsmith and C. M. Place. *An introduction to dynamical systems*. Cambridge, 1990.

[2] L. Bachmair and N. Dershowitz. Commutation, transformation, and termination. In J. H. Siekmann, editor, *Proc. 8th Int. Conf. on Automated Deduction*, volume 230 of *LNCS*, pages 5–20, Berlin, 1986. Springer-Verlag.

[3] V. D. Blondel, O. Bournez, P. Koiran, C. H. Papadimitriou, and J. N. Tsitsiklis. Deciding stability and mortality of piecewise affine dynamical system. *Theoretical Computer Science*, 255(1–2):687–696, 2001.

[4] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36:1249–1274, 2000.

[5] M. Colon and H. Sipma. Synthesis of linear ranking functions. In T. Margaria and W. Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 7th Intl. Conf. TACAS 2001*, volume 2031 of *LNCS*, pages 67–81. Springer, 2001.

[6] M. Colon and H. Sipma. Practical methods for proving program termination. In E. Brinksma and K. G. Larsen, editors, *Computer Aided Verification, 14th Intl. Conf. CAV 2002*, volume 2034 of *LNCS*, pages 442–454. Springer, 2002.

[7] D. Dams, R. Gerth, and O. Grumberg. A heuristic for the automatic generation of ranking function. In *Workshop on Advances in Verification WAVe 2000*, pages 1–8, 2000.

[8] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall PTR, 1997.

[9] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice-Hall, second edition, 1971.

[10] R. Kannan and R. J. Lipton. Polynomial-time algorithm for the orbit problem. *J. of the ACM*, 33(4):808–821, 1986.

[11] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *J. Symbolic Computation*, 32(3):231–253, 2001.

[12] J. Musset and M. Rusinowitch. Computing metatransitions for linear transition systems. In *12th Intl. FME symposium*, 2003.

[13] A. Podelski and A. Rybalchenko. A complete method for synthesis of linear ranking functions. In *VMCAI 2004: Verification, Model Checking, and Abstract Interpretation*, LNCS. Springer-Verlag, 2004.

[14] A. Podelski and A. Rybalchenko. Transition invariants. In *Logic in Computer Science, LICS*. IEEE Computer Society, 2004.

[15] D. R. Smart. *Fixed Point Theorems*. Cambridge University Press, 1980.

[16] E. Sontag. From linear to nonlinear: Some complexity comparisons. In *34th IEEE Conf. on Decision and Control, CDC*, 1995.

[17] A. Tiwari. Approximate reachability for linear systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, April 2003.