AProVE (KoAT + LoAT)

Nils Lommen, Florian Frohn, and Jürgen Giesl

Motivation

Goal: Prove or disprove termination of C programs

Motivation

Goal: Prove or disprove termination of C programs

```
while (x1 < x2) do
    [x1; x2] ← [3 · x1; 2 · x2]
end
```

$$\texttt{while}\ (x_1 < x_2)\ \texttt{do}\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix}\quad \texttt{end}$$

- Does this program terminate?

Motivation

Goal: Prove or disprove termination of C programs

```
while (x₁ < x₂ ∧ x₁ > 0) do
  [x₁; x₂] ← [3·x₁; 2·x₂]
end
```

$$\texttt{while}\ (x_1 < x_2 \land x_1 > 0)\ \texttt{do}$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix}$$
$$\texttt{end}$$

- Does this program terminate?

Motivation

Goal: Prove or disprove termination of C programs

```
while (x_1 < x_2 ∧ x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix}$$

```
end
```

- Does this program terminate?

Motivation

Goal: Prove or disprove termination of C programs

```
while (x₃ > 0) do



  while (x₁ < x₂ ∧ x₁ > 0) do
    [x₁, x₂] ← [3·x₁, 2·x₂]
  end
  [x₃] ← [x₃ − 1]
end
```

- Does this program terminate?

Motivation

Goal: Prove or disprove termination of C programs

$$
\begin{array}{l}
\texttt{while}\ (x_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix} \\
\quad \texttt{while}\ (x_1 < x_2 \land x_1 > 0)\ \texttt{do} \\
\qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Does this program terminate?

General Architecture

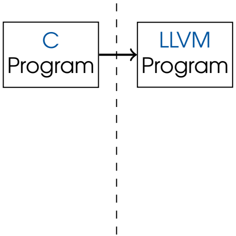
- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs

General Architecture

- APROVE (KoAT + LoAT) is a framework to analyze termination of C Programs

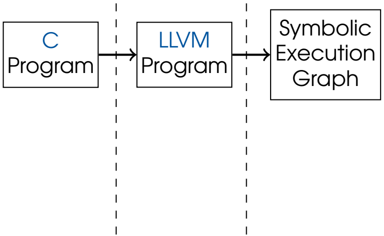
General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs



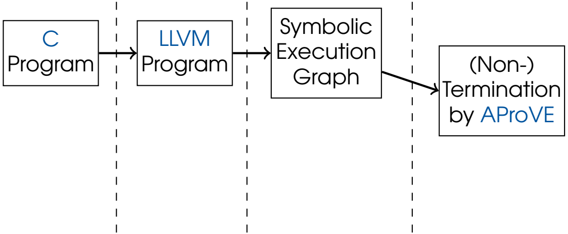
General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs



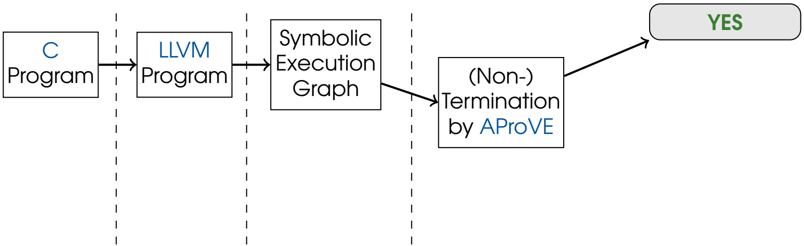
General Architecture

- APROVE (KoAT + LoAT) is a framework to analyze termination of C Programs



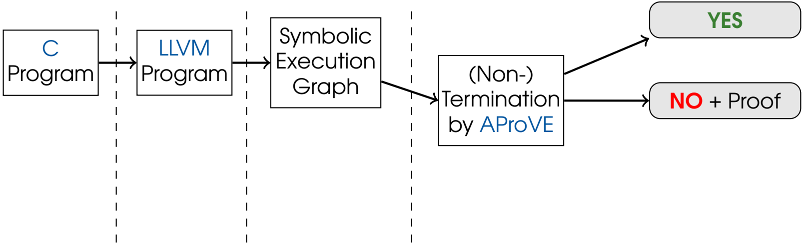
General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs



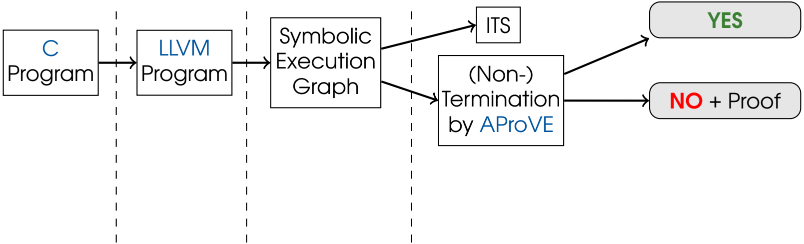
General Architecture

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs



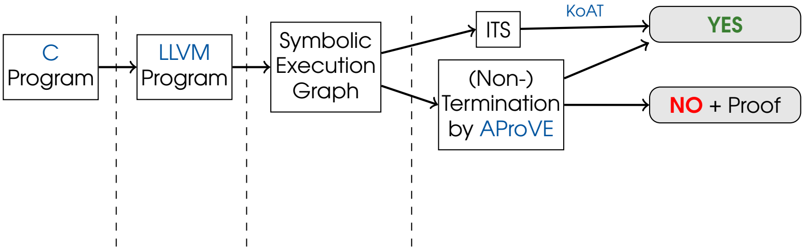
General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*



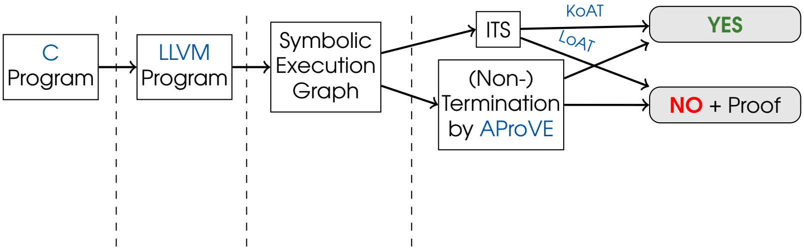
General Architecture

- APProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



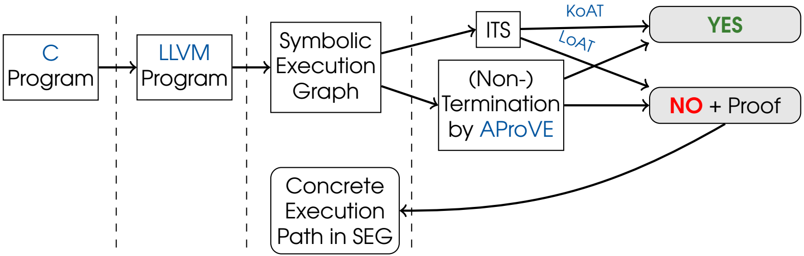
General Architecture

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



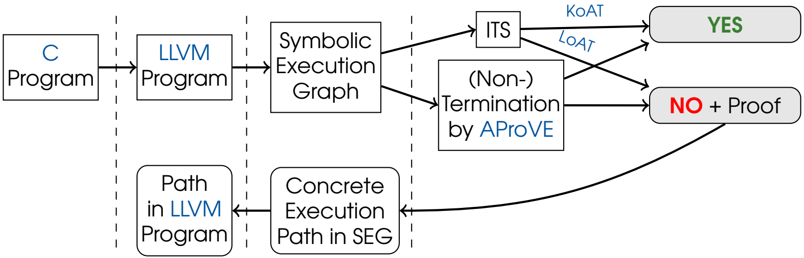
General Architecture

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



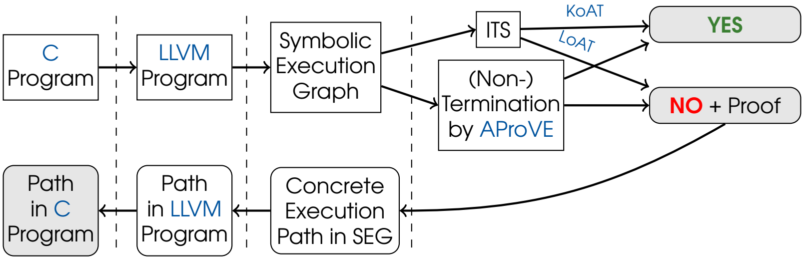
General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



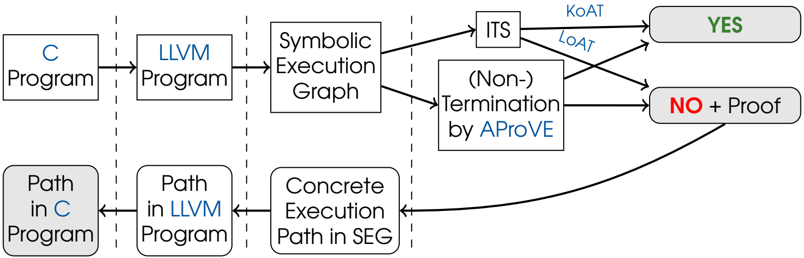
General Architecture

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



General Architecture

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



- KoAT – Ranking Functions
- KoAT – TWN-Loops

KoAT – Proving Termination via Ranking Functions

- Loops: $\texttt{while}\ (\varphi)\ \texttt{do}\ \eta\ \texttt{end}$

$$\texttt{while}\ (x_1 > 0)$$
$$\begin{bmatrix} x_1 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \end{bmatrix}$$
$$\texttt{end}$$

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Linear Ranking Functions

$$\texttt{while } (x_1 > 0)$$
$$[x_1] \leftarrow [x_1 - 1]$$
$$\texttt{end}$$

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Linear Ranking Functions
  - Search $\mathbf{a} \in \mathbb{Z}^{d+1}$ such that $f(\mathbf{a}, \mathbf{x}) = a_0 + a_1 x_1 + \cdots + a_d x_d$ yields well-founded order on $\mathbb{N}$

```
while (x1 > 0)
    [x1] ← [x1 − 1]
end
```

KoAT – Proving Termination via Ranking Functions

- Loops: $\texttt{while}\ (\varphi)\ \texttt{do}\ \eta\ \texttt{end}$
- Linear Ranking Functions
  - Search $\mathbf{a} \in \mathbb{Z}^{d+1}$ such that $f(\mathbf{a}, \mathbf{x}) = a_0 + a_1 x_1 + \cdots + a_d x_d$ yields well-founded order on $\mathbb{N}$
  - Decreasing:
  $$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f(\mathbf{a}, \mathbf{x}) \geq f(\mathbf{a}, \eta(\mathbf{x})) + 1$$

```
while (x₁ > 0)
    [x₁] ← [x₁ − 1]
end
```

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Linear Ranking Functions
  - Search $\mathbf{a} \in \mathbb{Z}^{d+1}$ such that $f(\mathbf{a}, \mathbf{x}) = a_0 + a_1 x_1 + \cdots + a_d x_d$ yields well-founded order on $\mathbb{N}$
  - Decreasing:
    $$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f(\mathbf{a}, \mathbf{x}) \geq f(\mathbf{a}, \eta(\mathbf{x})) + 1$$
  - Boundedness:
    $$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f(\mathbf{a}, \mathbf{x}) > 0$$

```
while (x_1 > 0)
    [x_1] ← [x_1 − 1]
end
```

KoAT – Proving Termination via Ranking Functions

- Loops: $\texttt{while} \ (\varphi) \ \texttt{do} \ \eta \ \texttt{end}$

```
while (x₁ > 0)
```

$$\texttt{while}\ (x_1 > 0)$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$
$$\texttt{end}$$

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)

$$\texttt{while}\ (x_1 > 0)$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$
$$\texttt{end}$$

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)
  - Search $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{Z}^{d+1}$ such that $f_1(\mathbf{a}_1, \mathbf{x}), \ldots, f_k(\mathbf{a}_k, \mathbf{x})$ yields well-founded order on $\mathbb{N}$

$$\texttt{while}\ (x_1 > 0)$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$
$$\texttt{end}$$

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)
  - Search $a_1, \dots, a_k \in \mathbb{Z}^{d+1}$ such that $f_1(a_1, \mathbf{x}), \dots, f_k(a_k, \mathbf{x})$ yields well-founded order on $\mathbb{N}$
  - Decreasing:

$$\forall \mathbf{x} \in \mathbb{Z}^d. \qquad \varphi \to f_1(a_1, \mathbf{x}) \geq f_1(a_1, \eta(\mathbf{x})) + 1$$

```
while (x₁ > 0)
```
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$
```
end
```

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)
  - Search $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{Z}^{d+1}$ such that $f_1(\mathbf{a}_1, \mathbf{x})$, $\ldots, f_k(\mathbf{a}_k, \mathbf{x})$ yields well-founded order on $\mathbb{N}$
  - Decreasing:

$$\forall \mathbf{x} \in \mathbb{Z}^d. \quad \varphi \to f_1(\mathbf{a}_1, \mathbf{x}) \geq f_1(\mathbf{a}_1, \eta(\mathbf{x})) + 1$$

$$\forall \mathbf{x} \in \mathbb{Z}^d. \forall i \in \{2, \ldots, k\}. \ \varphi \to f_i(\mathbf{a}_i, \mathbf{x}) + f_{i-1}(\mathbf{a}_{i-1}, \mathbf{x}) \geq f_i(\mathbf{a}_i, \eta(\mathbf{x})) + 1$$

`while` $(x_1 > 0)$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$

`end`

KoAT – Proving Termination via Ranking Functions

- Loops: `while (`φ`) do `η` end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)
  - Search $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathbb{Z}^{d+1}$ such that $f_1(\mathbf{a}_1, \mathbf{x})$, $\dots, f_k(\mathbf{a}_k, \mathbf{x})$ yields well-founded order on $\mathbb{N}$
  - Decreasing:

$$\forall \mathbf{x} \in \mathbb{Z}^d. \quad \varphi \to f_1(\mathbf{a}_1, \mathbf{x}) \geq f_1(\mathbf{a}_1, \eta(\mathbf{x})) + 1$$

$$\forall \mathbf{x} \in \mathbb{Z}^d. \forall i \in \{2, \dots, k\}. \ \varphi \to f_i(\mathbf{a}_i, \mathbf{x}) + f_{i-1}(\mathbf{a}_{i-1}, \mathbf{x}) \geq f_i(\mathbf{a}_i, \eta(\mathbf{x})) + 1$$

  - Boundedness:

$$\forall \mathbf{x} \in \mathbb{Z}^d. \ \varphi \to f_k(\mathbf{a}_k, \mathbf{x}) > 0$$

`while (`$x_1 > 0$`)`

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 - 1 \end{bmatrix}$$

`end`

Termination of Linear Programs

Ashish Tiwari*

SRI International
333 Ravenswood Ave, Menlo Park, CA, USA
tiwari@csl.sri.com

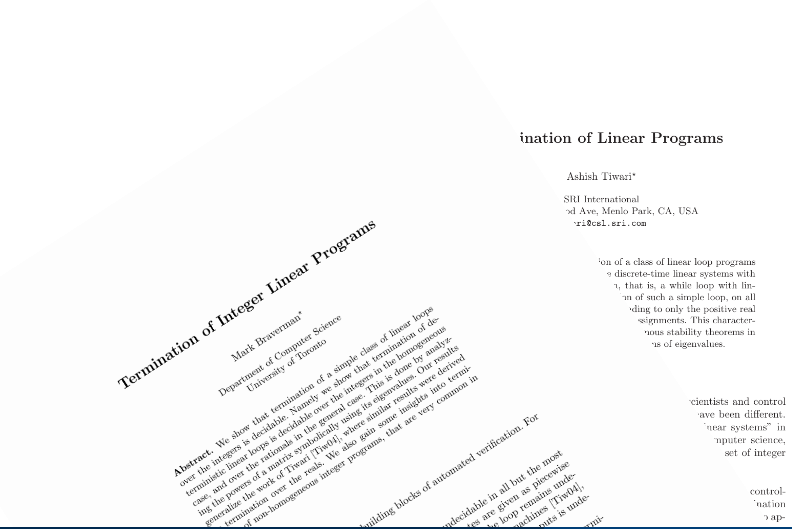
Abstract. We show that termination of a class of linear loop programs is decidable. Linear loop programs are discrete-time linear systems with a loop condition governing termination, that is, a while loop with linear assignments. We relate the termination of such a simple loop, on all initial values, 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 ap-



Termination of Integer Linear Programs

Mark Braverman*

Department of Computer Science
University of Toronto

Abstract. We show that termination of a simple class of linear loops over the integers is decidable. Namely we show that termination of deterministic linear loops is decidable over the integers in the homogeneous case, and over the rationals in the general case. This is done by analyzing the powers of a matrix symbolically using its eigenvalues. Our results generalize the work of Tiwari [Tiw04], where similar results were derived for termination over the reals. We also gain some insights into termination of non-homogeneous integer programs, that are very common in

...building blocks of automated verification. For
...ndecidable in all but the most
...tes are given as piecewise
...he loop remains unde-
...achines [Tiw04],
...puts is unde-
...rmi-

...ination of Linear Programs

Ashish Tiwari*

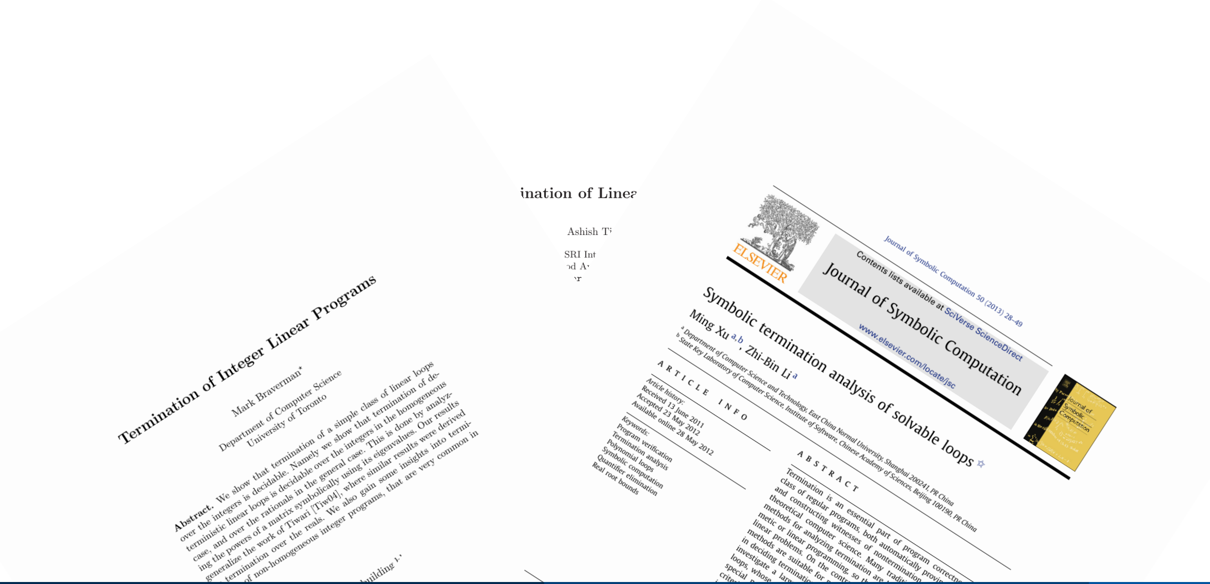
SRI International
...d Ave, Menlo Park, CA, USA
...ri@csl.sri.com

...ion of a class of linear loop programs
...e discrete-time linear systems with
...n, that is, a while loop with lin-
...on of such a simple loop, on all
...ding to only the positive real
...ssignments. This character-
...nous stability theorems in
...ns of eigenvalues.

...cientists and control
...ave been different.
...near systems" in
...mputer science,
set of integer

... control-
...uation
... ap-

KoAT – Analyzing Loops



Termination of Integer Linear Programs

Mark Braverman*

Department of Computer Science
University of Toronto

Abstract. We show that termination of a simple class of linear loops over the integers is decidable. Namely we show that termination of deterministic linear loops is decidable over the integers in the homogeneous case, and over the rationals in the general case. This is done by analyzing the powers of a matrix symbolically using its eigenvalues. Our results generalize the work of Tiwari [Tiw04], where similar results were derived for termination over the reals. We also gain some insights into termination of non-homogeneous integer programs, that are very common in

ination of Linea

Ashish Ti

SRI Int

Journal of Symbolic Computation 50 (2013) 28–49

Contents lists available at SciVerse ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

ELSEVIER

Symbolic termination analysis of solvable loops ☆

Ming Xu [a,b], Zhi-Bin Li [a]

[a] Department of Computer Science and Technology, East China Normal University, Shanghai 200241, PR China
[b] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, PR China

ARTICLE INFO

Article history:
Received 13 June 2011
Accepted 23 May 2012
Available online 28 May 2012

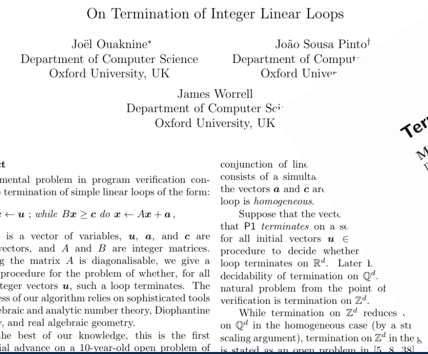
Keywords:
Program verification
Termination analysis
Polynomial loops
Symbolic computation
Quantifier elimination
Real root bounds

ABSTRACT

Termination is an essential part of program correctness ... class of regular programs, both automatically proving ... and constructing witnesses of nontermination ... theoretical computer science. Many tradi... methods for analyzing termination are ... metic or linear programming, so th... linear problems. On the contr... methods are suitable for n... in deciding termination... investigate a larg... loops, whose... special...

KoAT – Analyzing Loops

KoAT – Analyzing Loops



On Termination of Integer Linear Loops

Joël Ouaknine[*]
Department of Computer Science
Oxford University, UK

João Sousa Pinto[†]
Department of Comput…
Oxford Univer…

James Worrell
Department of Computer Sci…
Oxford University, UK

…t

…mental problem in program verification con… termination of simple linear loops of the form:

$… \leftarrow \boldsymbol{u}$; *while* $B\boldsymbol{x} \geq \boldsymbol{c}$ *do* $\boldsymbol{x} \leftarrow A\boldsymbol{x} + \boldsymbol{a}$,

… is a vector of variables, $\boldsymbol{u}$, $\boldsymbol{a}$, and $\boldsymbol{c}$ are …vectors, and A and B are integer matrices. …g the matrix A is diagonalisable, we give a … procedure for the problem of whether, for all …teger vectors $\boldsymbol{u}$, such a loop terminates. The …ess of our algorithm relies on sophisticated tools …ebraic and analytic number theory, Diophantine …, and real algebraic geometry.

…he best of our knowledge, this is the first …ial advance on a 10-year-old open problem of

conjunction of line… consists of a simulta… the vectors $\boldsymbol{a}$ and $\boldsymbol{c}$ ar… loop is *homogeneous*.

Suppose that the vect… that P1 *terminates* on a s… for all initial vectors $\boldsymbol{u}$ ∈ … procedure to decide whether … loop terminates on $\mathbb{R}^d$. Later … decidability of termination on $\mathbb{Q}^d$. … natural problem from the point of … verification is termination on $\mathbb{Z}^d$.

While termination on $\mathbb{Z}^d$ reduces … on $\mathbb{Q}^d$ in the homogeneous case (by a st… scaling argument), termination on $\mathbb{Z}^d$ in the … is stated as an open problem in [5, 8, 38] …

Termination of Linear Loops over the Integers

Mehran Hosseini
Department of Computer Science, University of Oxford, UK
mehran.hosseini@cs.ox.ac.uk

Joël Ouaknine
Max Planck Institute for Software Systems, Germany
Department of Computer Science, University of Oxford, UK
joel@mpi-sws.org

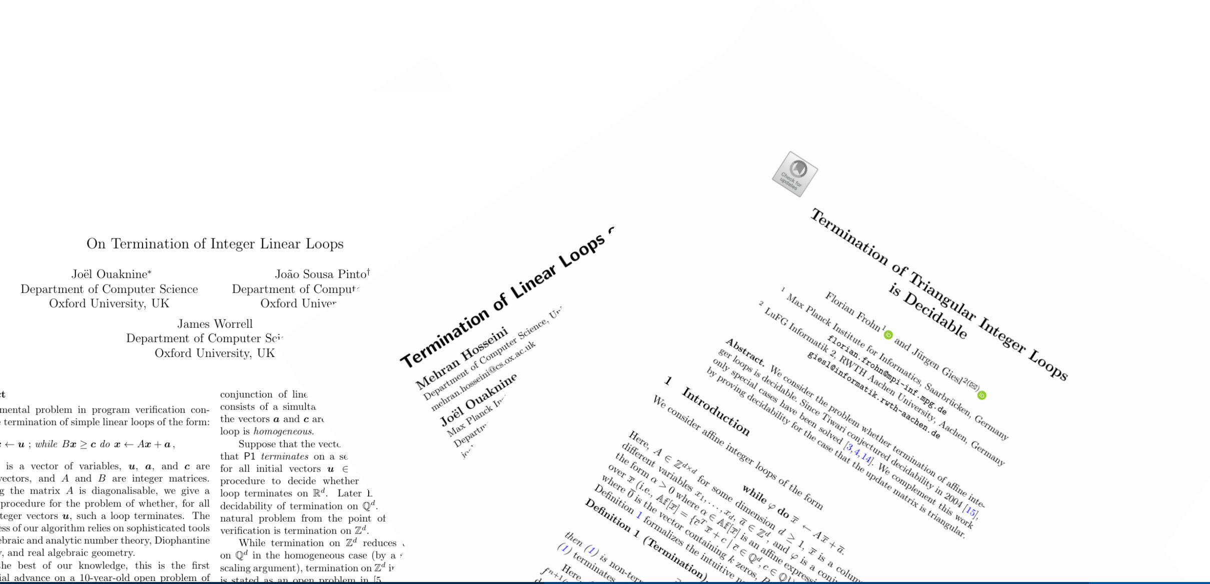
James Worrell
Department of Computer Science, University of Oxford, UK
jbw@cs.ox.ac.uk

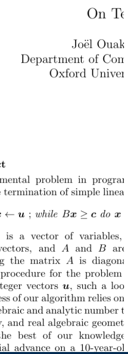
Abstract

We consider the problem of deciding termination of single-path while loops with integer variables, affine updates, and affine guard conditions. The question is whether such a loop terminates on all integer initial values. This problem is known to be decidable for the subclass of loops whose update matrices are diagonalisable, but the general case has remained open since being conjectured decidable by Tiwari in 2004. In this paper we show decidability of determining termination for arbitrary update matrices, confirming Tiwari's conjecture. For the class of loops considered in this …er, the question of deciding termination on a specific initial value is a longstanding open problem …er theory. The key to our decision procedure is in showing how to circumvent the difficulties … deciding termination on a fixed initial value.

…Classification Computing methodologies → Algebraic algorithms; Theory of …d verification
…Verification, Loop Termination, Linear Integer Programs, Affine
…LP.2019.118
…Theory of Programming
…by ERC grant AVS-ISS (648701).
…ISS (648701) and by DFG grant
…cience).
…208197/1.

KoAT – Analyzing Loops





On Te

Joël Ouak
Department of Con
Oxford Univer

t
mental problem in progra
termination of simple linea

$\leftarrow \boldsymbol{u}$; *while* $B\boldsymbol{x} \geq \boldsymbol{c}$ *do* $\boldsymbol{x}$

is a vector of variables,
vectors, and A and B are
g the matrix A is diagona
procedure for the problem
teger vectors $\boldsymbol{u}$, such a loo
ss of our algorithm relies on
ebraic and analytic number t
, and real algebraic geomet
he best of our knowledge
ial advance on a 10-year-ol

Check for updates

Termination of Polynomial Loops

Florian Frohn[1], Marcel Hark[2(✉)], and Jürgen Giesl[2]

[1] Max Planck Institute for Informatics and Saarland Informatics Campus, Saarbrücken, Germany
[2] LuFG Informatik 2, RWTH Aachen University, Aachen, Germany
marcel.hark@cs.rwth-aachen.de

Abstract. We consider the termination problem for triangular weakly non-linear loops (*twn*-loops) over some ring $\mathcal{S}$ like $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$. Essentially, the guard of such a loop is an arbitrary Boolean formula over (possibly non-linear) polynomial inequations, and the body is a single assignment $\begin{bmatrix} x_1 \\ \dots \\ x_d \end{bmatrix} \leftarrow \begin{bmatrix} c_1 \cdot x_1 + p_1 \\ \dots \\ c_d \cdot x_d + p_d \end{bmatrix}$ where each x_i is a variable, $c_i \in \mathcal{S}$, and each p_i is a (possibly non-linear) polynomial over $\mathcal{S}$ and the variables $x_{i+1}, \dots, x_d$.

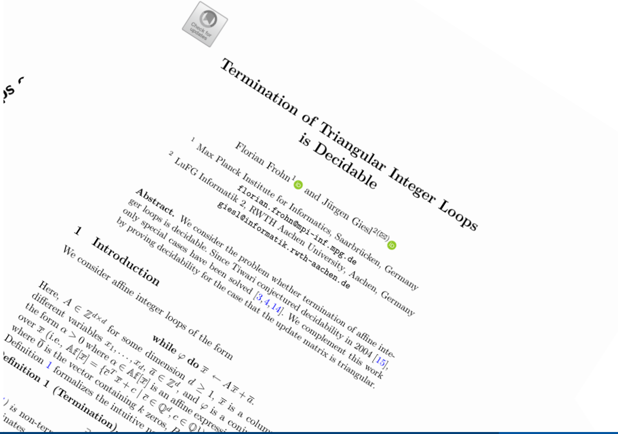
We present a reduction from the question of termination to the existential fragment of the first-order theory of $\mathcal{S}$ and $\mathbb{R}$. For loops over $\mathbb{R}$, our reduction entails decidability of termination. For loops over $\mathbb{Z}$ and $\mathbb{Q}$, it proves semi-decidability of non-termination.

Furthermore, we present a transformation to convert certain non-*twn*-loops into *twn*-form. Then the original loop terminates iff the transformed loop terminates over a specific subset of $\mathbb{R}$, which can also be checked via our reduction. This transformation also allows us to prove *tight* complexity bounds for the termination problem for two important classes of loops which can *always* be transformed into *twn*-loops.

1 Introduction

Let $\mathbb{R}_{\mathbb{A}}$ denote the real algebraic numbers. We consider loops of the form

$$\textbf{while } \varphi \textbf{ do } \vec{x} \leftarrow \vec{a}. \tag{1}$$



Check for updates

Termination of Triangular Integer Loops is Decidable

Florian Frohn[1] and Jürgen Giesl[2(✉)]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
florian.frohn@mpi-inf.mpg.de
[2] LuFG Informatik 2, RWTH Aachen University, Aachen, Germany
giesl@informatik.rwth-aachen.de

Abstract. We consider the problem whether termination of affine integer loops is decidable. Since Tiwari conjectured decidability in 2004 [15], only special cases have been solved [3,4,14]. We complement this work by proving decidability for the case that the update matrix is triangular.

1 Introduction

We consider affine integer loops of the form

while φ **do** $\overline{x} \leftarrow A\,\overline{x} + \overline{a}$.

Here, $A \in \mathbb{Z}^{d \times d}$ for some dimension $d \geq 1$, $\overline{x}$ is a colum
different variables $x_1, \dots, x_d$, $\overline{a} \in \mathbb{Z}^d$, and φ is a con
the form $\alpha > 0$ where $\alpha \in \mathbb{A}\mathrm{f}[\overline{x}]$ is an affine expressi
over $\overline{x}$ (i.e., $\mathbb{A}\mathrm{f}[\overline{x}] = \{\overline{c}^T \overline{x} + c \mid \overline{c} \in \mathbb{Q}^d, c \in \mathbb{Q}\}$
where $\overline{0}$ is the vector containing k zeros,
Definition 1 formalizes the intuitive n

efinition 1 (Termination).

Does the loop terminate?

$$\texttt{while } (\mathtt{x}_1 < \mathtt{x}_2 \land \mathtt{x}_1 > 0)$$

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot \mathtt{x}_1 \\ 2 \cdot \mathtt{x}_2 \end{bmatrix}$$

$$\texttt{end}$$

KoAT – Termination of TWN-Loops

Does the loop terminate?

$$\texttt{while}\ (x_1 < x_2 \land x_1 > 0)$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix}$$
$$\texttt{end}$$

- Yes!
- Value of x_1 eventually *outgrows* value of x_2

KoAT – Termination of TWN-Loops

Does the loop terminate?

```
while (x₁ < x₂ ∧ x₁ > 0)
    [x₁]    [3 · x₁]
    [x₂] ← [2 · x₂]
end
```

- Yes!
- Value of x_1 eventually *outgrows* value of x_2
- At some point we always have

$$3^n \cdot e_1 - 2^n \cdot e_2 \geq 0.$$

KoAT – Termination of TWN-Loops

Does the loop terminate?

```
while (x₁ < x₂ ∧ x₁ > 0)
    [x₁]  ←  [3 · x₁]
    [x₂]     [2 · x₂]
end
```

- Yes!
- Value of x_1 eventually *outgrows* value of x_2
- At some point we always have

$$3^n \cdot e_1 - 2^n \cdot e_2 \geq 0.$$

- Reduce Termination to an existential formula over $\mathbb{Z}$ (SAS '20)

KoAT – Termination of TWN-Loops

Does the loop terminate?

$$\texttt{while}\ (x_1 < x_2 \land x_1 > 0)$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix}$$
$$\texttt{end}$$

- Yes!
- Value of x_1 eventually *outgrows* value of x_2
- At some point we always have

$$3^n \cdot e_1 - 2^n \cdot e_2 \geq 0.$$

- Reduce Termination to an existential formula over $\mathbb{Z}$ (SAS '20)
  - linear arithmetic: co-NP-complete

Does the loop terminate?

```
while (x₁ < x₂ ∧ x₁ > 0)
    [x₁]    [3 · x₁]
    [x₂] ← [2 · x₂]
end
```

- Yes!
- Value of x_1 eventually *outgrows* value of x_2
- At some point we always have

$$3^n \cdot e_1 - 2^n \cdot e_2 \geq 0.$$

- Reduce Termination to an existential formula over $\mathbb{Z}$ (SAS '20)
  - linear arithmetic: co-NP-complete
  - non-linear arithmetic: non-termination is semi-decidable

Results of SV-Comp 2025

Termination:

Results of SV-Comp 2025

Termination:

- Proton: 3685 points

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

- AProVE (KoAT + LoAT) lags behind in categories *bit-vectors* and *other*

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

- AProVE (KoAT + LoAT) lags behind in categories *bit-vectors* and *other*
- witness missing (false(termination))

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

- AProVE (KoAT + LoAT) lags behind in categories *bit-vectors* and *other*
- witness missing (false(termination))
- floating points cannot be handled by AProVE (KoAT + LoAT)

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

- AProVE (KoAT + LoAT) lags behind in categories *bit-vectors* and *other*
- witness missing (false(termination))
- floating points cannot be handled by AProVE (KoAT + LoAT)
- investigate results in comparison to AProVE 2022

Challenges & Open Problems

- Challenges we came across:

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)
  - use newer clang and LLVM version

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)
  - use newer clang and LLVM version
  - improve certificate generation for non-termination

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)
  - use newer clang and LLVM version
  - improve certificate generation for non-termination
  - handle floating points

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)
  - use newer clang and LLVM version
  - improve certificate generation for non-termination
  - handle floating points
  - many more ...

Conclusion

- APROVE (KoAT + LoAT) is a framework to analyze termination of C Programs

Conclusion

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT

Conclusion

- APRoVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions

Conclusion

- APro VE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops

Conclusion

- APROVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops
- **Many** possible ways to improve AProVE (KoAT + LoAT)

Conclusion

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops
- **Many** possible ways to improve AProVE (KoAT + LoAT)

Check out our *poster* or *tool demo session* (Wednesday 2pm)

Conclusion

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops
- **Many** possible ways to improve AProVE (KoAT + LoAT)

Check out our *poster* or *tool demo session* (Wednesday 2pm)

`https://koat.verify.rwth-aachen.de/svcomp25`

Conclusion

- APro*V*E (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops
- **Many** possible ways to improve AProVE (KoAT + LoAT)

Check out our *poster* or *tool demo session* (Wednesday 2pm)

`https://koat.verify.rwth-aachen.de/svcomp25`

Thank You!