Polynomial Invariants for Affine Programs

Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, James Worrell

Max Planck Institute for Software Systems &
Department of Computer Science, Oxford University &
Mathematical Institute, Oxford University
Does this program halt?

**Affine program**

\[
x := 2^{-10} \\
y := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 2 & 0 \\ \frac{7}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

**Certificate of non-termination:**

\[
x^2 y - x^3 = 1023 - 1073741824 \tag{1}
\]

\[
y \xrightarrow{(1)} \text{ is an invariant: it holds at every step.}
\]
Does this program halt?

**Affine program**

\[
x := 2^{-10} \\
y := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 2 & 0 \\ \frac{7}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

Certificate of non-termination:

\[
x^2 y - x^3 = 1023/1073741824
\]

\(\vdash (1)\) is an invariant: it holds at every step

\(\vdash (1)\) implies the guard is true
Does this program halt?

**Affine program**

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} & := \begin{bmatrix} 2 & 0 \\ 7/4 & 1/4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

**Certificate of non-termination:**
\[
x^2 - x^3 = 1023 - 1073741824 (1)
\]

\(\text{\textbullet}\) is an invariant: it holds at every step

\(\text{\textbullet}\) implies the guard is true
Does this program halt?

Affine program

\[
x := 2^{-10}
\]
\[
y := 1
\]
while \( y \geq x \) do
\[
\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 2 & 0 \\ \frac{7}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

Certificate of non-termination:
\[
x^2 - y - 2^{10} = 1023 = 1073741824\footnote{(1)}
\]

\[
\text{\footnote{(1)} is an invariant: it holds at every step.}
\]
Does this program halt?

**Affine program**

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \quad & \left[
\begin{array}{l}
x \\
y
\end{array}
\right] := \left[
\begin{array}{cc}
2 & 0 \\
\frac{7}{4} & \frac{1}{4}
\end{array}
\right] \left[
\begin{array}{l}
x \\
y
\end{array}
\right]
\end{align*}
\]

Certificate of non-termination:
\[
x^2 - x^3 = 1023 - 1073741824
\]
Does this program halt?

**Affine program**

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} & := \begin{bmatrix} 2 & 0 \\ \frac{7}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

Certification of non-termination:

\[
x^2 y - x^3 = 1023 - 1073741824
\]

\(\text{(1)}\) is an invariant: it holds at every step.

\(\text{Guard}\) implies the guard is true.
Does this program halt?

**Affine program**

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} & := \begin{bmatrix} 2 & 0 \\ \frac{7}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

**Certificate of non-termination:**

\[
x^2y - x^3 = \frac{1023}{1073741824} \tag{1}
\]
Does this program halt?

Affine program

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} & := \begin{bmatrix} 2 & 0 \\ 7/4 & 1/4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

Certificate of non-termination:

\[
x^2 y - x^3 = \frac{1023}{1073741824} \tag{1}
\]

\[\begin{align*}
\blacktriangleright & \text{ (1) is an invariant: it holds at every step}
\end{align*}\]
Does this program halt?

Affine program

\[
\begin{align*}
x & := 2^{-10} \\
y & := 1 \\
\text{while } y \geq x \text{ do} \\
\begin{bmatrix} x \\ y \end{bmatrix} & := \\n\begin{bmatrix} 2 & 0 \\ 7/4 & 1/4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

Certificate of non-termination:

\[x^2y - x^3 = \frac{1023}{1073741824} \quad (1)\]

- (1) is an invariant: it holds at every step
- (1) implies the guard is true
Invariants

invariant = overapproximation of the reachable states
**Invariants**

**Invariant** = overapproximation of the reachable states

**Inductive invariant** = invariant preserved by the transition relation
Inductive invariants: example
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \to \mathbb{R}^3$
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

\[ f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
Inductive invariants: example

\(x, y, z\) range over \(\mathbb{Q}\)

\[f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3\]
Inductive invariants: example

Let $x, y, z$ range over $\mathbb{Q}$.

The functions $f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ are defined as:

- $f_1$ and $f_2$ connect nodes 1 and 2.
- $f_3$ and $f_4$ connect nodes 2 and 3.
- $f_5$ connects node 1 back to node 1.

The diagram illustrates the flow of these functions between the nodes.
Inductive invariants: example

\[ x, y, z \text{ range over } \mathbb{Q} \]

\[ f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]
Inductive invariants: example

\[ x, y, z \text{ range over } \mathbb{Q} \]

\[ f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]
Inductive invariants: example

\[ x, y, z \text{ range over } \mathbb{Q} \]

\[ f_i : \mathbb{R}^3 \to \mathbb{R}^3 \]
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$

$S_1, S_2, S_3$ is an invariant
Inductive invariants: example

\[ x, y, z \text{ range over } \mathbb{Q} \]

\[ f_i : \mathbb{R}^3 \to \mathbb{R}^3 \]

\[ S_1, S_2, S_3 \text{ is an inductive invariant} \]
Inductive invariants: example

\( x, y, z \) range over \( \mathbb{Q} \)

\[ f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]

\( l_1, l_2, l_3 \) is an invariant
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \to \mathbb{R}^3$

$l_1, l_2, l_3$ is NOT an inductive invariant
Inductive invariants: example

$x, y, z$ range over $\mathbb{Q}$

$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$

$l_1, l_2, l_3$ is an inductive invariant
Why Invariants?

The classical approach to the verification of temporal safety properties of programs requires the construction of inductive invariants [...]. Automation of this construction is the main challenge in program verification.

D. Beyer, T. Henzinger, R. Majumdar, and A. Rybalchenko
Invariant Synthesis for Combined Theories, 2007
Which invariants?

Octagons

Polyhedrons

Affine/linear sets

Algebraic sets = polynomial equalities

Semialgebraic sets
Which invariants?

Intervals

Octagons

Polyhedrons

Affine/linear sets

Algebraic sets = polynomial equalities

Semialgebraic sets
Which invariants?

Octagons

\[ \vee \]

Intervals
Which invariants?

Octagons

\[ \vee \]

Intervals

Affine/linear sets
Which invariants?

Intervals

Octagons

\lor

Polyhedrons

\lor

\lor

Intervals

Affine/linear sets

Algebraic sets = polynomial equalities

Semialgebraic sets

\leq \frac{6}{17}
Which invariants?

- Octagons
- Polyhedrons
- Intervals
- Affine/linear sets
- Algebraic sets = polynomial equalities
Which invariants?

- Octagons
- Polyhedrons
- Semialgebraic sets
- Intervals
- Affine/linear sets
- Algebraic sets = polynomial equalities
Affine programs

- Nondeterministic branching (no guards)
- All assignments are affine
- Allow nondeterministic assignments ($x := \ast$)
- Can overapproximate complex programs
- Covers existing formalisms: probabilistic, quantum, quantitative automata
Affine programs

- Nondeterministic branching (no guards)

![Diagram of nondeterministic branching](image)
Affine programs

- Nondeterministic branching (no guards)
- All assignments are affine

\[ x := 3x - 7y + 1 \]
Affine programs

- Nondeterministic branching (no guards)
- All assignments are affine
- Allow nondeterministic assignments ($x := *$)

\[ x := 3x - 7y + 1 \]

\[ y := * \]
Affine programs

- Nondeterministic branching (no guards)
- All assignments are affine
- Allow nondeterministic assignments ($x := *$)

Can overapproximate complex programs
Affine programs

- Nondeterministic branching (no guards)
- All assignments are affine
- Allow nondeterministic assignments \((x := \ast)\)

\[
x := 3x - 7y + 1
\]

- Can overapproximate complex programs
- Covers existing formalisms: probabilistic, quantum, quantitative automata
Theorem (Karr 76)

There is an algorithm which computes, for any given affine program over $\mathbb{Q}$, its strongest affine inductive invariant.
Discovering Affine Equalities Using Random Interpretation

Sumit Gulwani  George C. Necula  
University of California, Berkeley  
{gulwani, necula}@cs.berkeley.edu

**ABSTRACT**

We present a new polynomial-time randomized algorithm for discovering affine equalities involving variables in a program.

**Keywords**

Affine Relationships, Linear Equalities, Random Interpretation, Randomized Algorithm
Some polynomial invariants

A Note on Karr’s Algorithm

Markus Müller-Olm¹* and Helmut Seidl²

Abstract. We give a simple formulation of Karr’s algorithm for computing all affine relationships in affine programs. This simplified algorithm runs in time $O(nk^3)$ where $n$ is the program size and $k$ is the number of program variables assuming unit cost for arithmetic operations. This improves upon the original formulation by a factor of $k$. Moreover, our re-formulation avoids exponential growth of the lengths of intermediately occurring numbers (in binary representation) and uses less complicated elementary operations. We also describe a generalization that determines all polynomial relations up to degree $d$ in time $O(nk^{3d})$.

Theorem (ICALP 2004)

*There is an algorithm which computes, for any given affine program over $\mathbb{Q}$, all its polynomial inductive invariants up to any fixed degree $d$.\*
A challenge: finding all polynomial invariants

Available online at www.sciencedirect.com

SCIENCE DIRECT

Information Processing Letters 91 (2004) 233–244

ELSEVIER

Information Processing Letters

www.elsevier.com/locate/ipl

Computing polynomial program invariants

Markus Müller-Olm a,*,1, Helmut Seidl b

a FernUniversität Hagen, LG Praktische Informatik 5, 58084 Hagen, Germany
b TU München, Informatik, I2, 85748 München, Germany

Received 16 October 2003; received in revised form 20 April 2004
Available online 19 June 2004
Computing polynomial program invariants

Markus Müller-Olm a,*, 1, Helmut Seidl b

a FernUniversität Hagen, LG Praktische Informatik 5, 58084 Hagen, Germany
b TU München, Informatik, I2, 85748 München, Germany

Received 16 October 2003; received in revised form 20 April 2004
Available online 19 June 2004

It is a challenging open problem whether or not the set of all valid polynomial relations can be computed not just the ones of some given form. It is not
Why fixed degree is not enough

Paraboloid

\[ z = x^2 + y^2 \]

Union of 3 hyperplanes

\[(x - y)(10y + x)(y + 10x) = 0 \]
Why fixed degree is not enough

- Paraboloid

\[ z = x^2 + y^2 \]
Why fixed degree is not enough

- Paraboloid

\[ z = x^2 + y^2 \]
Why fixed degree is not enough

- Paraboloid
- Union of 3 hyperplanes

\[ z = x^2 + y^2 \]

\[(x - y)(10y + x)(y + 10x) = 0\]
Why fixed degree is not enough

- Paraboloid
  \[ z = x^2 + y^2 \]
- Union of 3 hyperplanes
  \[ (x - y)(10y + x)(y + 10x) = 0 \]
Why fixed degree is not enough

- Paraboloid
  \[ z = x^2 + y^2 \]
- Union of 3 hyperplanes
  \[ (x - y)(10y + x)(y + 10x) = 0 \]
Main result

Theorem

There is an algorithm which computes, for any given affine program over $\mathbb{Q}$, its strongest polynomial inductive invariant.
Main result

Theorem

*There is an algorithm which computes, for any given affine program over \( \mathbb{Q} \), its strongest polynomial inductive invariant.*

- strongest polynomial invariant \( \iff \) smallest algebraic set
Main result

**Theorem**

*There is an algorithm which computes, for any given affine program over \( \mathbb{Q} \), its strongest polynomial inductive invariant.*

- **strongest polynomial invariant** \( \iff \) **smallest algebraic set**
  - **algebraic sets** = finite \( \bigcup \) and \( \bigcap \) of polynomial equalities
Main result

**Theorem**

*There is an algorithm which computes, for any given affine program over \( \mathbb{Q} \), its strongest polynomial inductive invariant.*

- **strongest polynomial invariant** \( \iff \) **smallest algebraic set**
  - **algebraic sets** = finite \( \bigcup \) and \( \bigcap \) of polynomial equalities

- Thus our algorithm computes **all polynomial relations** that always hold among program variables at each program location, in all possible executions of the program
Main result

Theorem

There is an algorithm which computes, for any given affine program over \( \mathbb{Q} \), its strongest polynomial inductive invariant.

- strongest polynomial invariant \( \iff \) smallest algebraic set
  - algebraic sets = finite \( \cup \) and \( \cap \) of polynomial equalities

Thus our algorithm computes all polynomial relations that always hold among program variables at each program location, in all possible executions of the program.

- We can represent this (usually infinite) set of relations using a finite basis of polynomial equalities
At the edge of decidability

\[ x := M_1 x \]
\[ x := M_2 x \]
\[ x := M_k x \]

\[ x := x_0 \]

\[ S \]

Theorem (Markov 1947): There is a fixed set of \(6 \times 6\) integer matrices \(M_1, \ldots, M_k\) such that the reachability problem "y is reachable from \(x_0\)?" is undecidable.

Theorem (Paterson 1970): The mortality problem "0 is reachable from \(x_0\) with \(M_1, \ldots, M_k\)" is undecidable for \(3 \times 3\) matrices.

\(*\) Original theorems about semigroups, reformulated with affine programs.
At the edge of decidability

\[ x := M_1 x \]

\[ x := x_0 \]

\[ x := M_2 x \]

\[ \ldots \]

\[ x := M_k x \]

**Theorem (Markov 1947*)**

*There is a fixed set of 6 \times 6 integer matrices \( M_1, \ldots, M_k \) such that the reachability problem “\( y \) is reachable from \( x_0 \)” is undecidable.*

*Original theorems about semigroups, reformulated with affine programs.*
At the edge of decidability

\[ x := M_1 x \]
\[ x := M_2 x \]
\[ x := M_3 x \]
\[ \ldots \]
\[ x := M_k x \]

\[ S \]

**Theorem (Markov 1947\(^*\))**

There is a **fixed set** of \( 6 \times 6 \) integer matrices \( M_1, \ldots, M_k \) such that the reachability problem “\( y \) is reachable from \( x_0 \)?” is **undecidable**.

**Theorem (Paterson 1970\(^*\))**

The mortality problem “\( 0 \) is reachable from \( x_0 \) with \( M_1, \ldots, M_k \)?” is **undecidable** for \( 3 \times 3 \) matrices.

\(^*\)Original theorems about semigroups, reformulated with affine programs.
Tools

- Algebraic geometry
- Number theory
- Group theory
Theorem (Derksen, Jeandel and Koiran, 2004)

There is an algorithm which computes, for any given affine program over $\mathbb{Q}$ using only invertible transformations, its strongest polynomial inductive invariant.
Main contribution

Theorem

*Given a finite set of rational square matrices of the same dimension, we can compute the Zariski closure of the semigroup that they generate.*

Corollary

*Given an affine program, we can compute for each location the ideal of all polynomial relations that hold at that location.*
Summary

- invariant = overapproximation of reachable states
- invariants allow verification of safety properties
- affine program:
  - nondeterministic branching, no guards, affine assignments

Theorem

There is an algorithm which computes, for any given affine program over \( \mathbb{Q} \), its strongest polynomial inductive invariant.