# Program Invariants

James Worrell

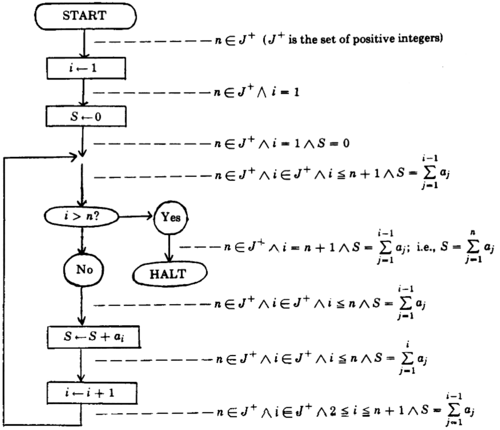
Department of Computer Science, Oxford University

(joint work with Ehud Hrushovski, Joël Ouaknine, Amaury Pouly)

WST 2018, Oxford
July 2018

# Programming in the Jurassic

destination (or origin) is $v$. An *interpretation* $I$ of a flowchart is a mapping of its edges on propositions. Some, but not necessarily all, of the free variables of these propositions may be variables manipulated by the



START
$n \in J^+$ ($J^+$ is the set of positive integers)
$i \leftarrow 1$
$n \in J^+ \wedge i = 1$
$S \leftarrow 0$
$n \in J^+ \wedge i = 1 \wedge S = 0$
$n \in J^+ \wedge i \in J^+ \wedge i \leqq n+1 \wedge S = \sum_{j=1}^{i-1} a_j$
$i > n?$
Yes
$n \in J^+ \wedge i = n+1 \wedge S = \sum_{j=1}^{i-1} a_j$; i.e., $S = \sum_{j=1}^{n} a_j$
No
HALT
$n \in J^+ \wedge i \in J^+ \wedge i \leqq n \wedge S = \sum_{j=1}^{i-1} a_j$
$S \leftarrow S + a_i$
$n \in J^+ \wedge i \in J^+ \wedge i \leqq n \wedge S = \sum_{j=1}^{i} a_j$
$i \leftarrow i + 1$
$n \in J^+ \wedge i \in J^+ \wedge 2 \leqq i \leqq n+1 \wedge S = \sum_{j=1}^{i-1} a_j$

FIGURE 1. Flowchart of program to compute $S = \sum_{j=1}^{n} a_j$ ($n \geqq 0$)

Robert W. Floyd, *Assigning Meanings to Programs*, 1967

## Automated Invariant Generation: A Modern Challenge

*The classical approach to the verification of temporal safety properties of programs requires the construction of* **inductive invariants** *at each program point, that is, assertions that are true on every program execution reaching that point, and moreover, that are closed under the strongest postcondition operator.* **Automation of this construction is the main challenge in program verification.**

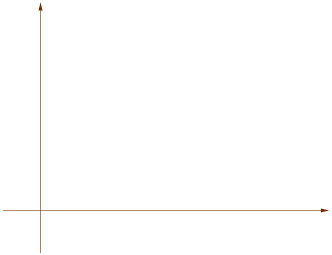
D. Beyer, T. Henzinger, R. Majumdar, and A. Rybelchenko
*Invariant Synthesis for Combined Theories*, 2007

## Does This Loop Terminate?

```
$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do
        $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$
```

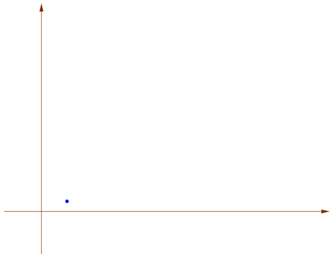
## Does This Loop Terminate?

```
$x := 3;$
$y := 2;$
while  $2y - x \geq -2$  do
    $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$
```



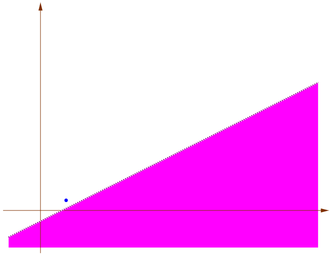
## Does This Loop Terminate?

```
$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do
    $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$;
```



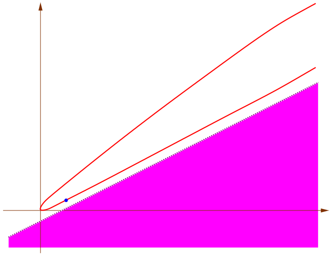
## Does This Loop Terminate?

```
$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do
    $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$
```



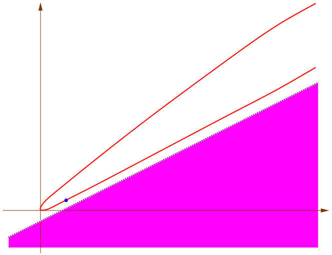
## Does This Loop Terminate?

```
$x := 3$;
$y := 2$;
while $2y - x \geq -2$ do
    $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$;
```



## Does This Loop Terminate?

```
$x := 3;$
$y := 2;$
while  $2y - x \geq -2$  do
    $\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$
```



Polynomial invariant: $x - 9x^2 - y + 24xy - 16y^2 = 0$

## Termination of Linear Loops

**Linear-Loop Termination**

Given a vector $\boldsymbol{x} \in \mathbb{Q}^d$, a halfspace $F \subseteq \mathbb{Q}^d$ and a matrix $A \in \mathbb{Q}^{d \times d}$, does there there exist $n \in \mathbb{N}$ such that $A^n \boldsymbol{x} \in F$?

## Termination of Linear Loops

**Linear-Loop Termination**

Given a vector $\boldsymbol{x} \in \mathbb{Q}^d$, a halfspace $F \subseteq \mathbb{Q}^d$ and a matrix $A \in \mathbb{Q}^{d \times d}$, does there there exist $n \in \mathbb{N}$ such that $A^n \boldsymbol{x} \in F$?

- Open for many decades!

# Termination of Linear Loops

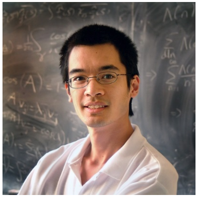
**Linear-Loop Termination**

Given a vector $\boldsymbol{x} \in \mathbb{Q}^d$, a halfspace $F \subseteq \mathbb{Q}^d$ and a matrix $A \in \mathbb{Q}^{d \times d}$, does there there exist $n \in \mathbb{N}$ such that $A^n \boldsymbol{x} \in F$?

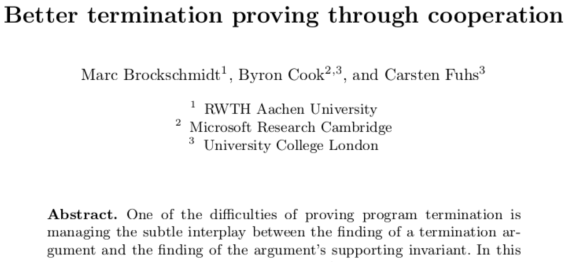
- Open for many decades!

*"It is faintly outrageous that this problem is still open; it is saying that we do not know how to decide the Halting Problem even for 'linear' automata!"*

Terence Tao



# Invariants Can (help to) Prove Liveness



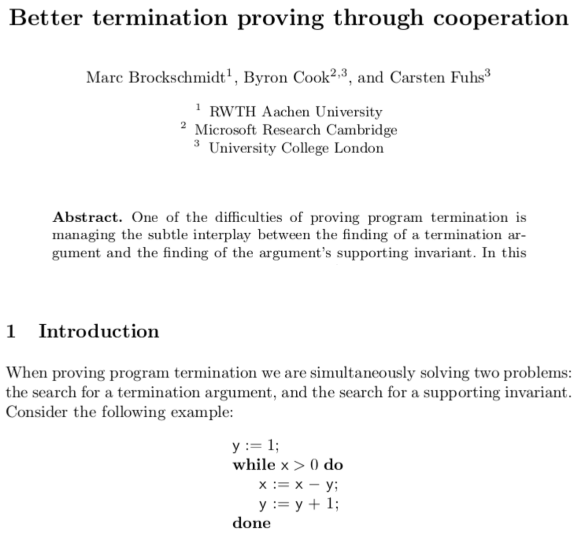
**Better termination proving through cooperation**

Marc Brockschmidt[1], Byron Cook[2,3], and Carsten Fuhs[3]

[1] RWTH Aachen University
[2] Microsoft Research Cambridge
[3] University College London

**Abstract.** One of the difficulties of proving program termination is managing the subtle interplay between the finding of a termination argument and the finding of the argument's supporting invariant. In this

## Invariants Can (help to) Prove Liveness



# Better termination proving through cooperation

Marc Brockschmidt[1], Byron Cook[2,3], and Carsten Fuhs[3]

[1] RWTH Aachen University
[2] Microsoft Research Cambridge
[3] University College London

**Abstract.** One of the difficulties of proving program termination is managing the subtle interplay between the finding of a termination argument and the finding of the argument's supporting invariant. In this

## 1 Introduction

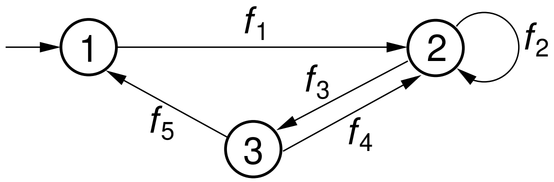
When proving program termination we are simultaneously solving two problems: the search for a termination argument, and the search for a supporting invariant. Consider the following example:

```
y := 1;
while x > 0 do
    x := x − y;
    y := y + 1;
done
```

## From Flowcharts to Affine Programs

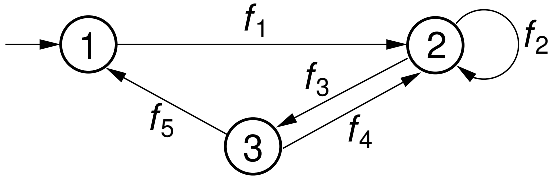
$$\begin{tikzcd}
{} \arrow[r] & 1 \arrow[rr, "f_1"] & & 2 \arrow[loop right, "f_2"] \arrow[dl, bend right=10, "f_3"'] \\
& & 3 \arrow[ul, "f_5"] \arrow[ur, bend right=10, "f_4"'] &
\end{tikzcd}$$

## From Flowcharts to Affine Programs



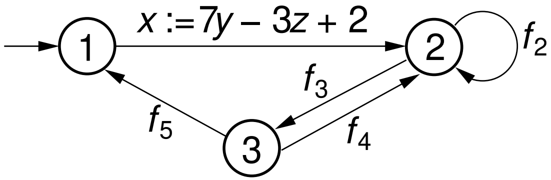
- Only ‘nondeterministic’ branching (no conditionals)

## From Flowcharts to Affine Programs



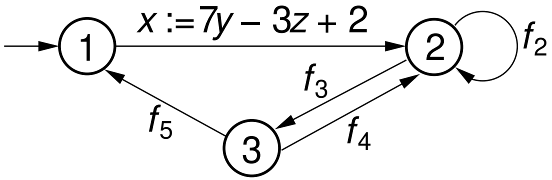
- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine

## From Flowcharts to Affine Programs



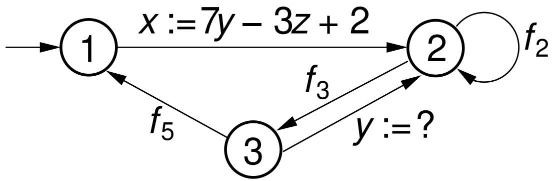
- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine

## From Flowcharts to Affine Programs



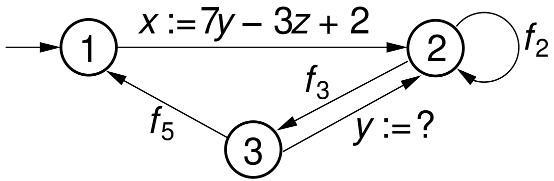
- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine
- Also allow nondeterministic assignments $x :=\ ?$

## From Flowcharts to Affine Programs


$$\begin{tikzcd} {} \arrow[r] & 1 \arrow[rr, "x := 7y - 3z + 2"] & & 2 \arrow[loop right, "f_2"] \arrow[dl, bend left, "f_3"] \\ & & 3 \arrow[ul, "f_5"] \arrow[ur, bend left, "y := ?"'] & \end{tikzcd}$$

- Only 'nondeterministic' branching (no conditionals)
- All assignments are affine
- Also allow nondeterministic assignments $x := ?$

## From Flowcharts to Affine Programs

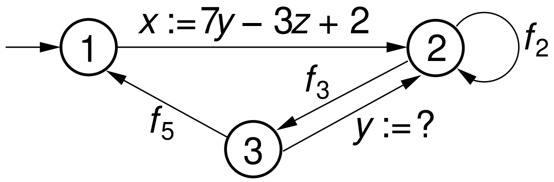


- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine
- Also allow nondeterministic assignments $x := ?$

Affine programs:

- can **overapproximate** more complex programs

## From Flowcharts to Affine Programs



- Only 'nondeterministic' branching (no conditionals)
- All assignments are affine
- Also allow nondeterministic assignments $x := ?$

Affine programs:

- can **overapproximate** more complex programs
- already cover a range of existing formalisms, e.g. **probabilistic** and **quantum automata**, ...

## Inductive Invariants

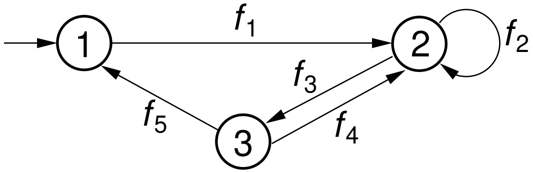
**invariant** = overapproximation (of the reachable states)

## Inductive Invariants

**invariant** = overapproximation (of the reachable states)

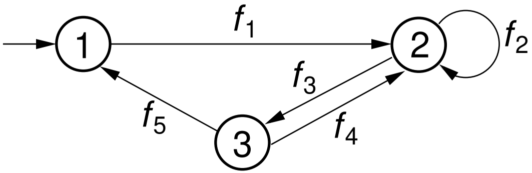
$$\textbf{inductive invariant} = \begin{cases} \text{overapproximation} \\ \text{preserved by the transition relation} \end{cases}$$

## Inductive Invariants



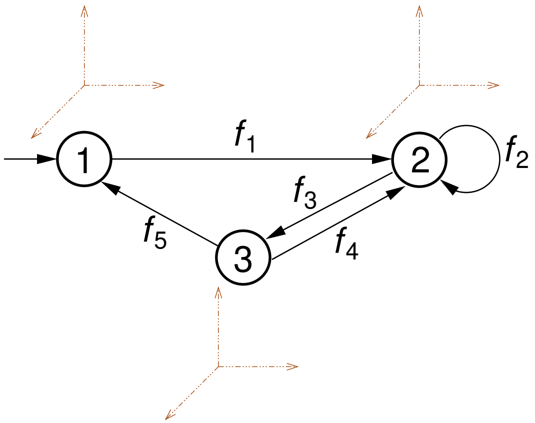
## Inductive Invariants

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



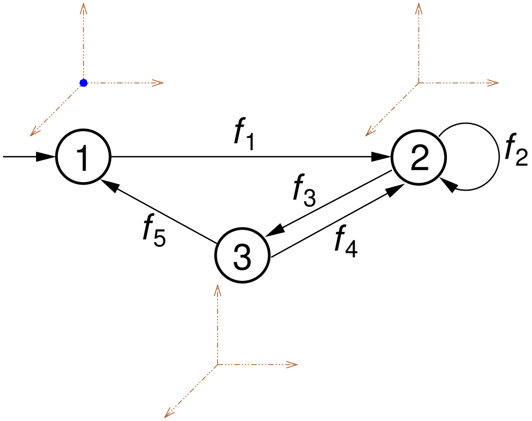
## Inductive Invariants

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



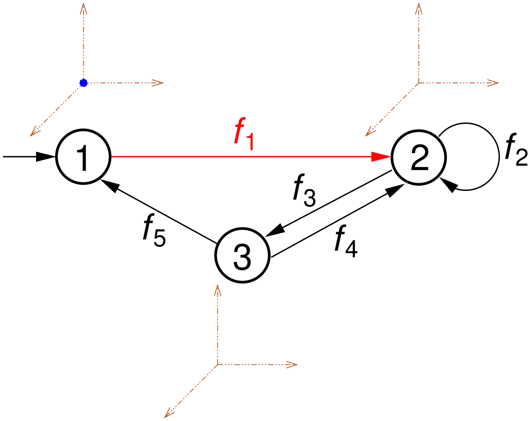
## Inductive Invariants

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



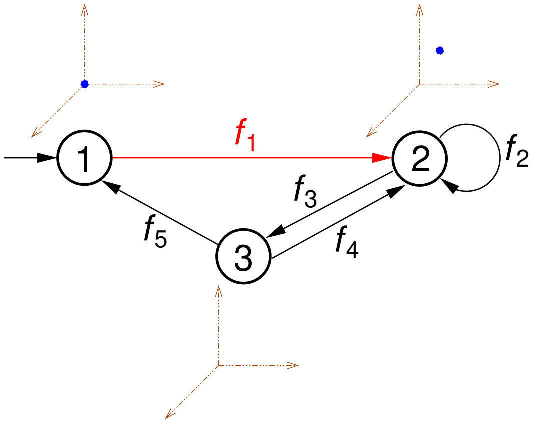
## Inductive Invariants

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



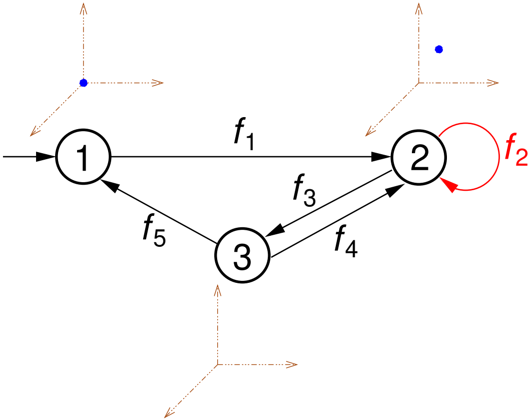
## Inductive Invariants

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



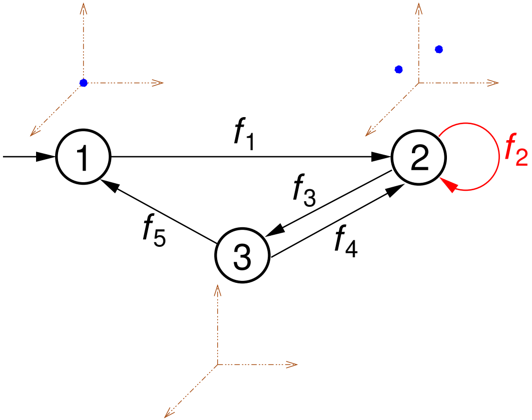
## Inductive Invariants

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



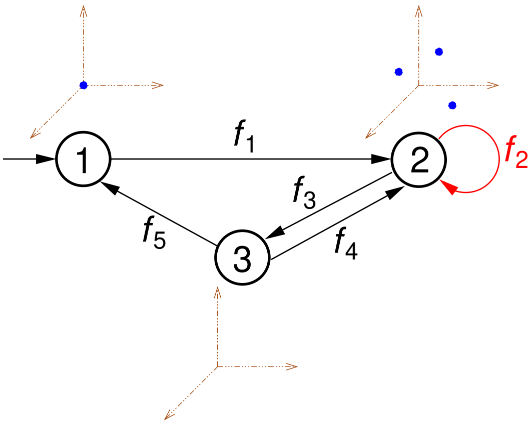
## Inductive Invariants

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



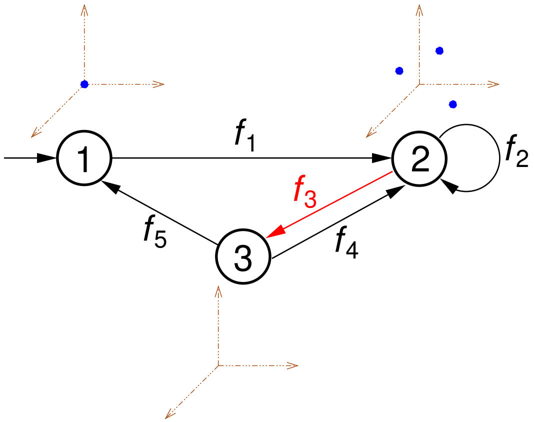
## Inductive Invariants

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



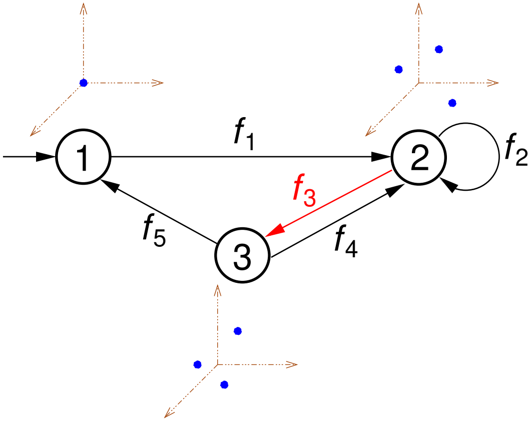
## Inductive Invariants

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



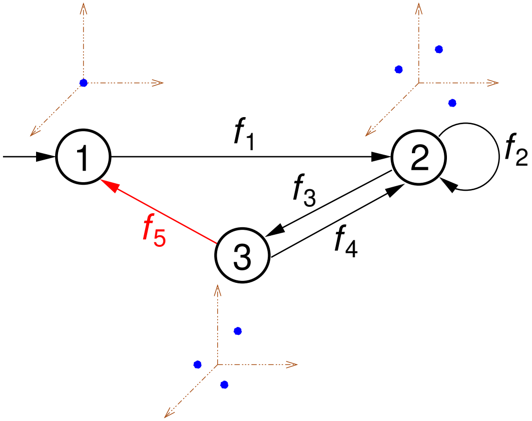
## Inductive Invariants

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



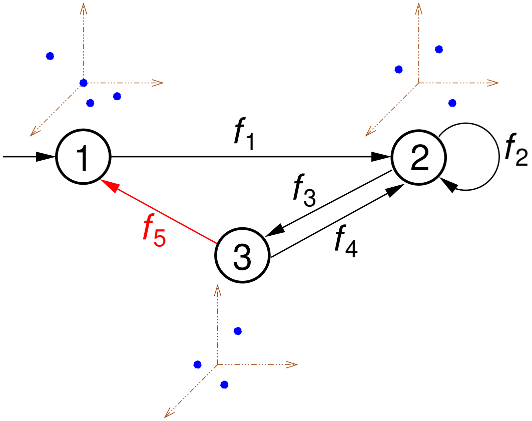
## Inductive Invariants

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



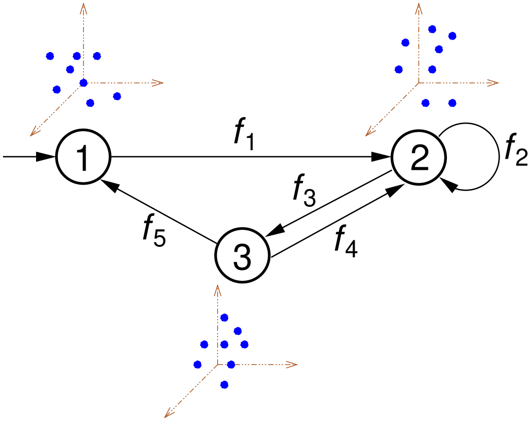
## Inductive Invariants

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



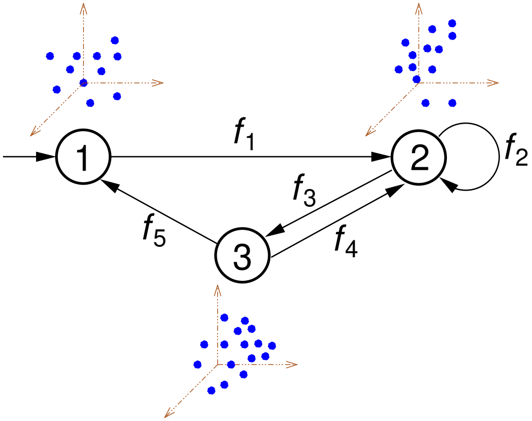
## Inductive Invariants

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



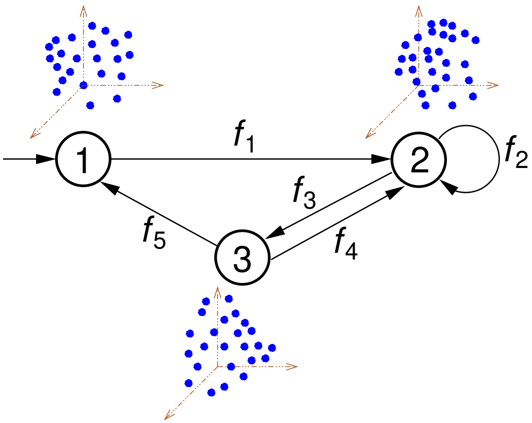
## Inductive Invariants

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



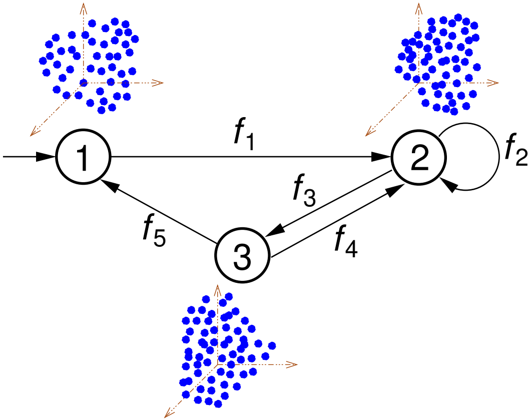
## Inductive Invariants

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



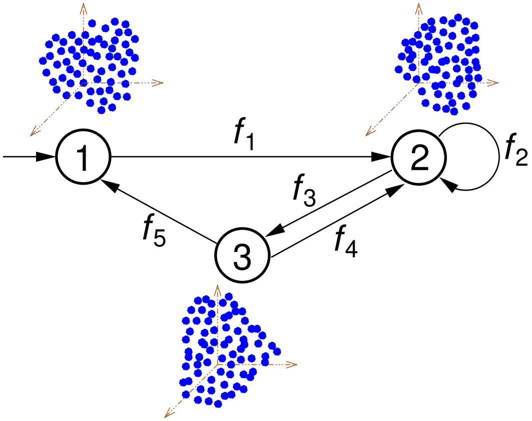
## Inductive Invariants

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



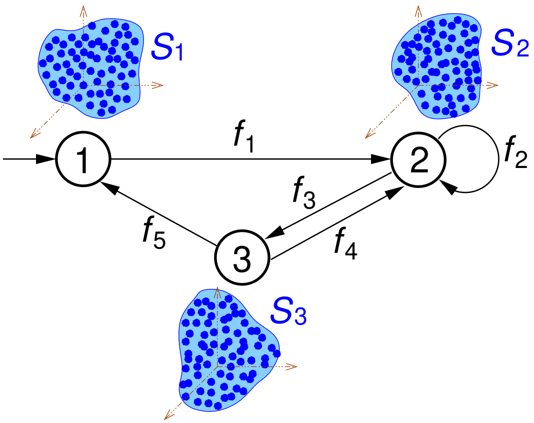
## Inductive Invariants

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



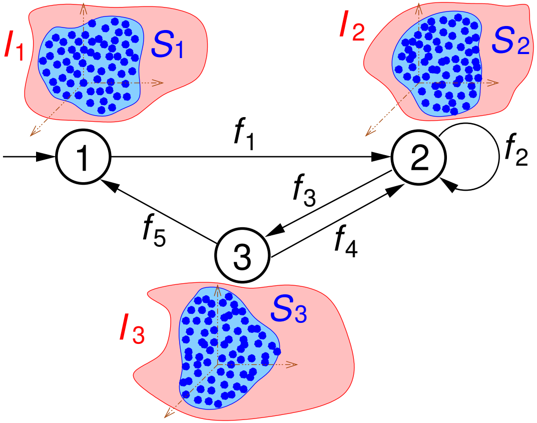
## Inductive Invariants

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



## Inductive Invariants

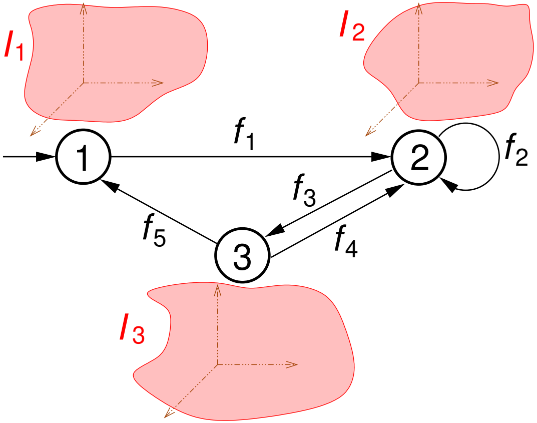
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **invariant** ($I_1, I_2, I_3 \subseteq \mathbb{R}^3$)

## Inductive Invariants

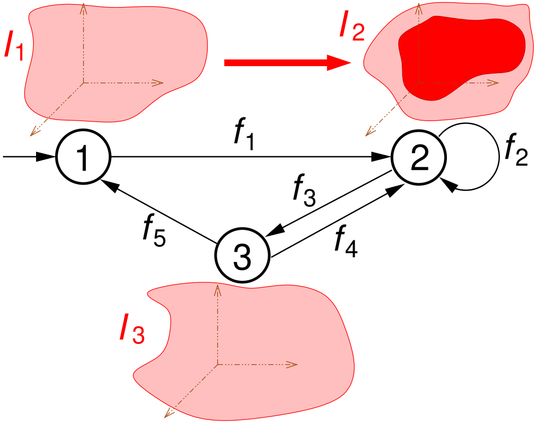
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **invariant** ($I_1, I_2, I_3 \subseteq \mathbb{R}^3$)

## Inductive Invariants

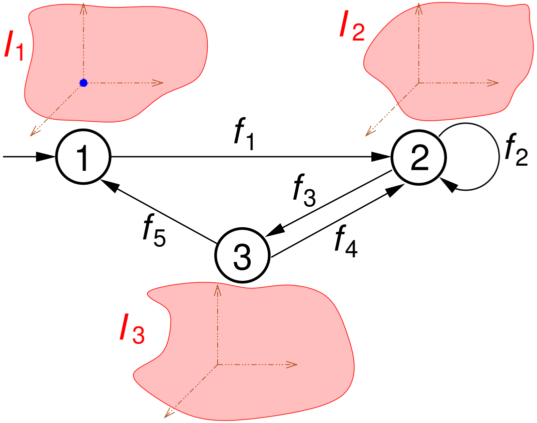
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **inductive invariant** ($I_1, I_2, I_3 \subseteq \mathbb{R}^3$)

## Inductive Invariants

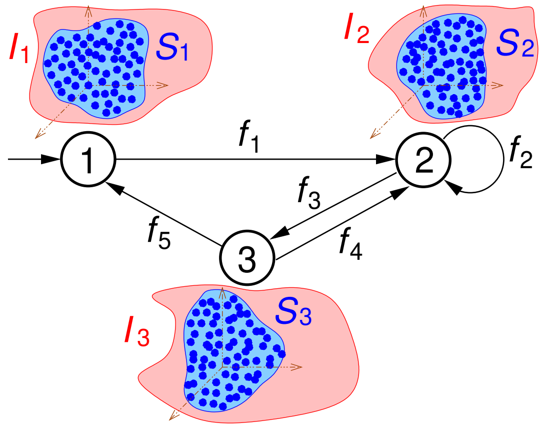
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **inductive invariant** ($I_1, I_2, I_3 \subseteq \mathbb{R}^3$)

## Inductive Invariants

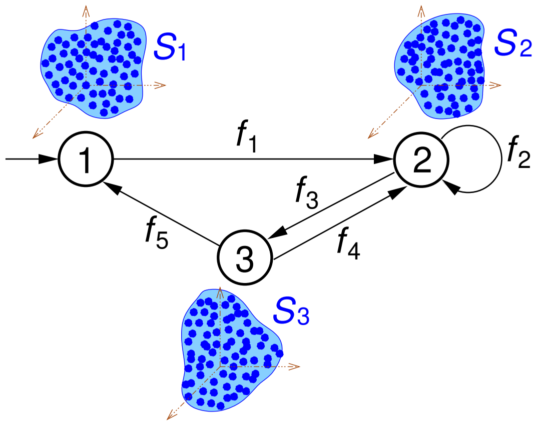
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **inductive invariant** ($I_1, I_2, I_3 \subseteq \mathbb{R}^3$)

## Inductive Invariants

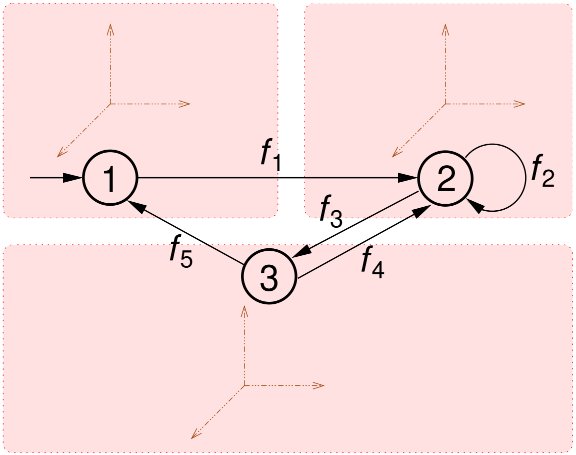
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle S_1, S_2, S_3 \rangle$ is **always** an inductive invariant

## Inductive Invariants

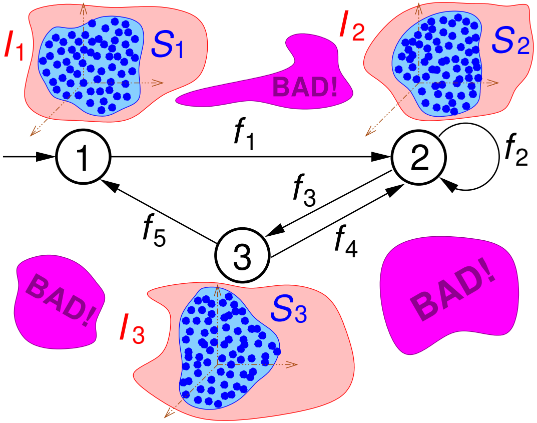
$x, y, z$ range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3 \rangle$ is also **always** an inductive invariant

## Inductive Invariants

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



A good invariant is worth a thousand reachability queries!

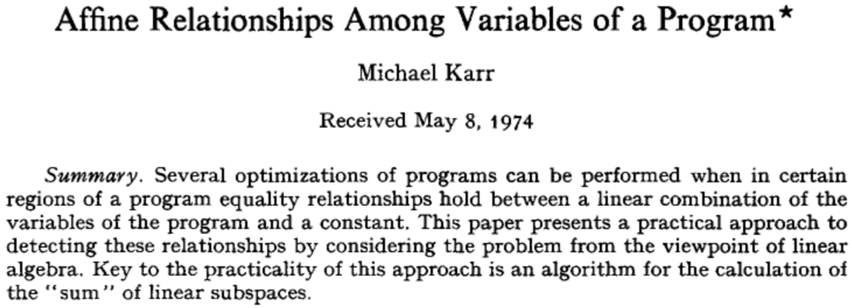
## Generating Inductive Invariants

- Choose the right abstract domain
  - Some domains always have ‘best’ (strongest, smallest) invariants, others not

## Generating Inductive Invariants

- Choose the right abstract domain
  - Some domains always have ‘best’ (strongest, smallest) invariants, others not
- Compute an invariant!
  - Many eclectic methods: fixed-point computations, constraint solving, interpolation, abduction, machine learning, . . .
  - Some approaches require 'widening’ to ensure termination
  - Other techniques invoke e.g. dimension or algebraic arguments
  - Trade-off between precision and tractability . . .

## Karr's Algorithm, Acta Informatica 76



Affine Relationships Among Variables of a Program*

Michael Karr

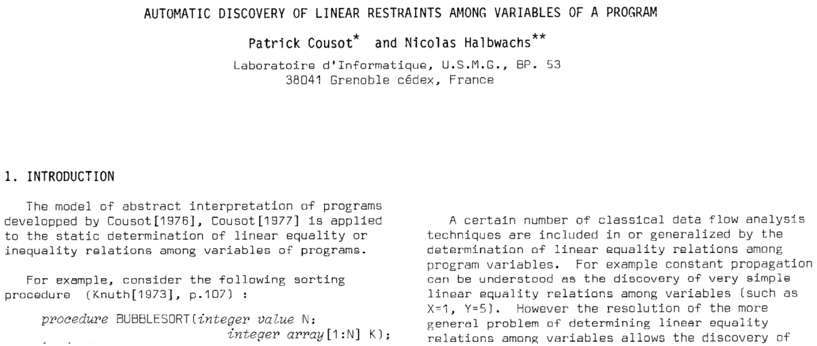
Received May 8, 1974

*Summary*. Several optimizations of programs can be performed when in certain regions of a program equality relationships hold between a linear combination of the variables of the program and a constant. This paper presents a practical approach to detecting these relationships by considering the problem from the viewpoint of linear algebra. Key to the practicality of this approach is an algorithm for the calculation of the "sum" of linear subspaces.

**Theorem (Karr 76)**

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

# Polyhedral Invariants @ POPL 1978



AUTOMATIC DISCOVERY OF LINEAR RESTRAINTS AMONG VARIABLES OF A PROGRAM

Patrick Cousot* and Nicolas Halbwachs**

Laboratoire d'Informatique, U.S.M.G., BP. 53
38041 Grenoble cédex, France

1. INTRODUCTION

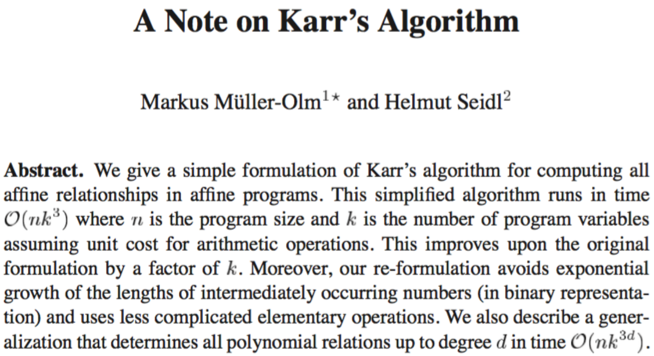
The model of abstract interpretation of programs developped by Cousot[1976], Cousot[1977] is applied to the static determination of linear equality or inequality relations among variables of programs.

For example, consider the following sorting procedure (Knuth[1973], p.107) :

```
    procedure BUBBLESORT(integer value N;
                                  integer array[1:N] K);
```

A certain number of classical data flow analysis techniques are included in or generalized by the determination of linear equality relations among program variables. For example constant propagation can be understood as the discovery of very simple linear equality relations among variables (such as X=1, Y=5). However the resolution of the more general problem of determining linear equality relations among variables allows the discovery of

## Karr Goes Polynomial @ ICALP 2004



### A Note on Karr's Algorithm

Markus Müller-Olm[1*] and Helmut Seidl[2]

**Abstract.** We give a simple formulation of Karr's algorithm for computing all affine relationships in affine programs. This simplified algorithm runs in time $\mathcal{O}(nk^3)$ 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 $\mathcal{O}(nk^{3d})$.

## Why Linear Invariants Are Not Enough

```
$s := 0$;
$x := 0$;
while $\langle \ldots \rangle$ do
        $x := x + 1$;
        $s := s + x$;
```

## Why Linear Invariants Are Not Enough

```
$s := 0$;
$x := 0$;
while $\langle \ldots \rangle$ do
    $x := x + 1$;
    $s := s + x$;
```

The loop invariant is:

$$s = \frac{x(x+1)}{2}$$

## Why Linear Invariants Are Not Enough

```
$s := 0;$
$x := 0;$
while $\langle \dots \rangle$ do
    $x := x + 1;$
    $s := s + x;$
```

The loop invariant is:

$$s = \frac{x(x+1)}{2}$$

Or equivalently:

$$p(s, x) = 2s - x^2 - x = 0$$

## More on Polynomial Invariants

[Sankaranarayanan, Sipma, and Manna 04]
[Ildikó, Kovács, and T. Jebelean 05],
[Müller-Olm, Potter, Seidl 06],
[Rodríguez-Carbonell and Kapur 07a],
[Rodríguez-Carbonell and Kapur 07b],
[Colón 07],
[Kovács 08],
[Kapur 13],
[Cachera, Jensen, Jobin, and Kirchner 14],
[de Oliveira, Bensalem, and Prevosto 16],
[Humenberger, Jaroschek, L. Kovács 18],
...

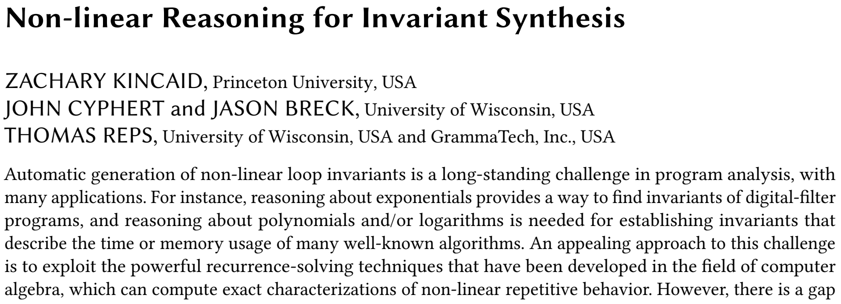
## More on Polynomial Invariants

[Sankaranarayanan, Sipma, and Manna 04]
[Ildikó, Kovács, and T. Jebelean 05],
[Müller-Olm, Potter, Seidl 06],
[Rodríguez-Carbonell and Kapur 07a],
[Rodríguez-Carbonell and Kapur 07b],
[Colón 07],
[Kovács 08],
[Kapur 13],
[Cachera, Jensen, Jobin, and Kirchner 14],
[de Oliveira, Bensalem, and Prevosto 16],
[Humenberger, Jaroschek, L. Kovács 18],
...

**Tools:**

- *Aligator:* Gröbner bases
- *Fastind:* constraints-based; avoids Gröbner bases
- *Pilat:* linearisation

## Non-Linear Invariants @ POPL 2018

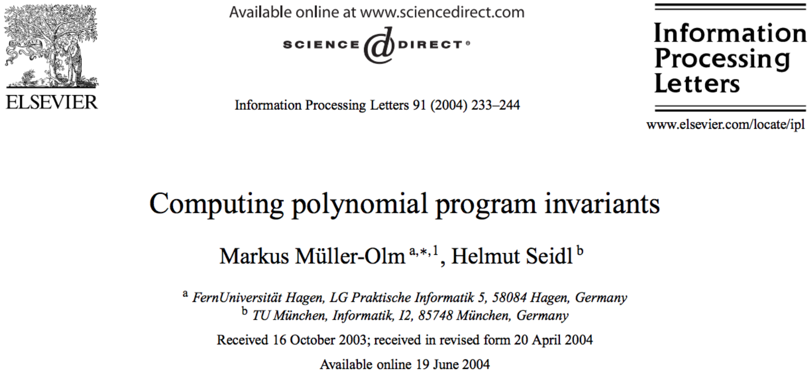


# Non-linear Reasoning for Invariant Synthesis

ZACHARY KINCAID, Princeton University, USA
JOHN CYPHERT and JASON BRECK, University of Wisconsin, USA
THOMAS REPS, University of Wisconsin, USA and GrammaTech, Inc., USA

Automatic generation of non-linear loop invariants is a long-standing challenge in program analysis, with many applications. For instance, reasoning about exponentials provides a way to find invariants of digital-filter programs, and reasoning about polynomials and/or logarithms is needed for establishing invariants that describe the time or memory usage of many well-known algorithms. An appealing approach to this challenge is to exploit the powerful recurrence-solving techniques that have been developed in the field of computer algebra, which can compute exact characterizations of non-linear repetitive behavior. However, there is a gap

## A Challenge on Polynomial Invariants @ IPL 2004



Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 91 (2004) 233–244

Information Processing Letters

www.elsevier.com/locate/ipl

ELSEVIER

Computing polynomial program invariants

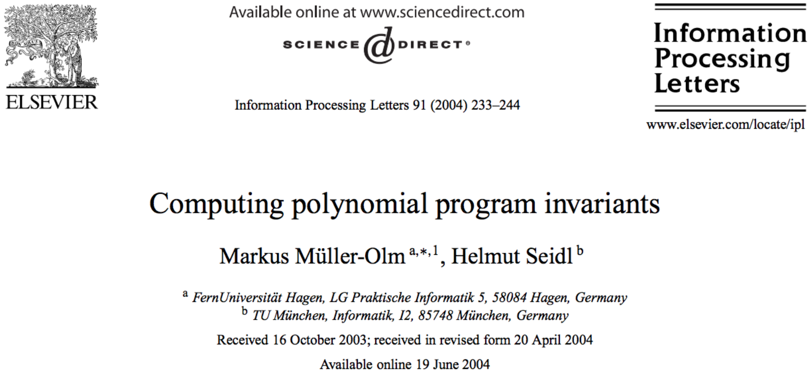
Markus Müller-Olm [a,*,1], Helmut Seidl [b]

[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

# A Challenge on Polynomial Invariants @ IPL 2004



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 91 (2004) 233–244

Information Processing Letters

www.elsevier.com/locate/ipl

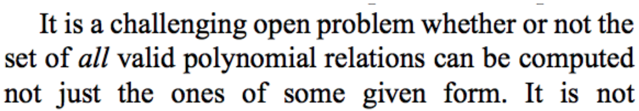
Computing polynomial program invariants

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



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

## Strongest Polynomial Invariants @ LICS 2018

Theorem (Hrushovski, Ouaknine, Pouly, W. 18)

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

## Strongest Polynomial Invariants @ LICS 2018

**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

*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

## Strongest Polynomial Invariants @ LICS 2018

**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

*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 are defined by conjunctions of polynomial equalities

## Strongest Polynomial Invariants @ LICS 2018

**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

*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 are defined by conjunctions of polynomial equalities
- Algorithm computes for each program location the set of **all** polynomial relations that hold among program variables at whenever control reaches that location

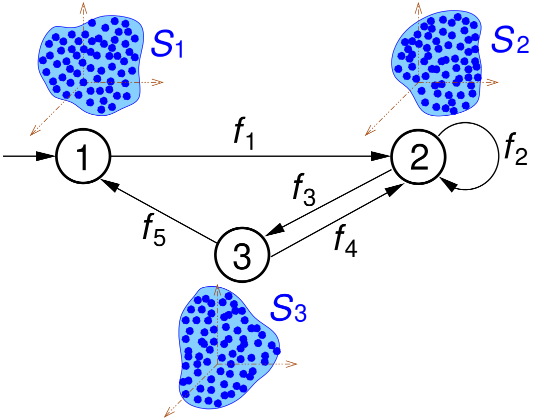
## Strongest Polynomial Invariants @ LICS 2018

**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

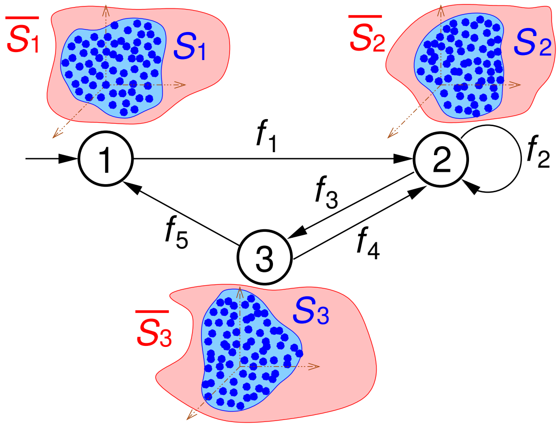
*There is an algorithm which computes, for any given affine program over $\mathbb{Q}$, its strongest polynomial inductive invariant.*

- strongest polynomial invariant $\iff$ smallest algebraic set
    - algebraic sets are defined by conjunctions of polynomial equalities
- Algorithm computes for each program location the set of **all** polynomial relations that hold among program variables at whenever control reaches that location
- We represent this set of relations using a **finite basis** of polynomial equalities

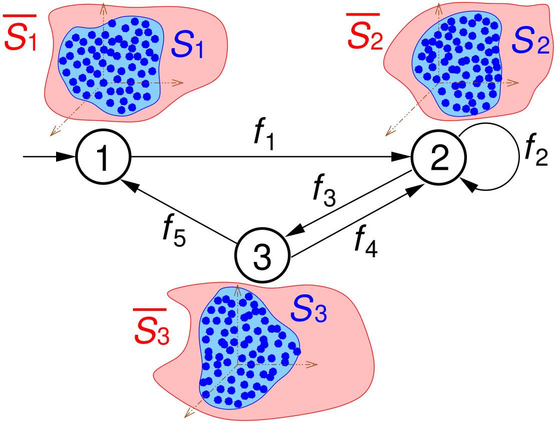
## Computing Strongest Inductive Polynomial Invariants



## Computing Strongest Inductive Polynomial Invariants

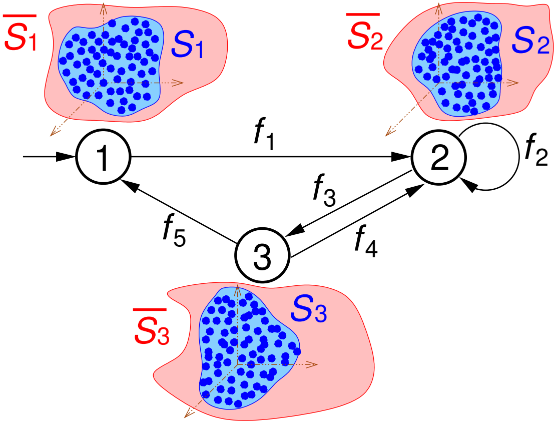


# Computing Strongest Inductive Polynomial Invariants



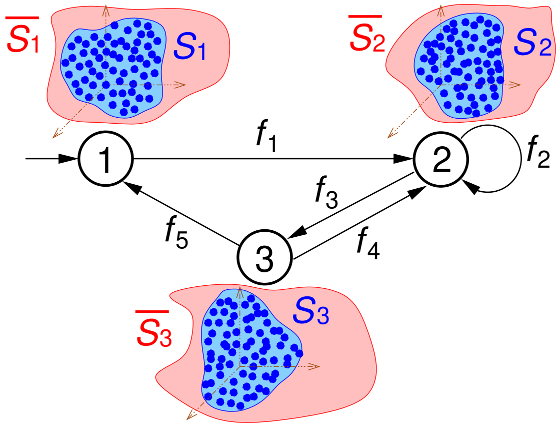
$f_1(S_1) \subseteq S_2$

## Computing Strongest Inductive Polynomial Invariants



$$f_1(S_1) \subseteq S_2 \quad \Longrightarrow \quad f_1(\overline{S_1}) \subseteq \overline{S_2}$$

# Computing Strongest Inductive Polynomial Invariants



$$f_1(S_1) \subseteq S_2 \quad \Longrightarrow \quad f_1(\overline{S_1}) \subseteq \overline{S_2}$$

…because affine functions are Zariski-continuous

## Algebraic Sets & the Zariski Topology

Algebraic sets are closed under:

- finite unions (by taking products of polynomials)

## Algebraic Sets & the Zariski Topology

Algebraic sets are closed under:

- finite unions (by taking products of polynomials)
- finite intersections (by definition)

## Algebraic Sets & the Zariski Topology

Algebraic sets are closed under:

- finite unions (by taking products of polynomials)
- finite intersections (by definition)
- **arbitrary** intersections (by Hilbert Basis Theorem)

$$\mathbf{V}(\mathcal{P}) = \mathbf{V}(p_1, \ldots, p_k) \text{ for some } p_1, \ldots, p_k \in \mathcal{P}$$

## Algebraic Sets & the Zariski Topology

Algebraic sets are closed under:

- finite unions (by taking products of polynomials)
- finite intersections (by definition)
- **arbitrary** intersections (by Hilbert Basis Theorem)

$$\mathbf{V}(\mathcal{P}) = \mathbf{V}(p_1, \ldots, p_k) \text{ for some } p_1, \ldots, p_k \in \mathcal{P}$$

$$\Longrightarrow$$

*algebraic sets* $\equiv$ *closed sets* in the **Zariski topology**

## Algebraic Sets & the Zariski Topology

Algebraic sets are closed under:

- finite unions (by taking products of polynomials)
- finite intersections (by definition)
- **arbitrary** intersections (by Hilbert Basis Theorem)

$$\mathbf{V}(\mathcal{P}) = \mathbf{V}(p_1, \dots, p_k) \text{ for some } p_1, \dots, p_k \in \mathcal{P}$$

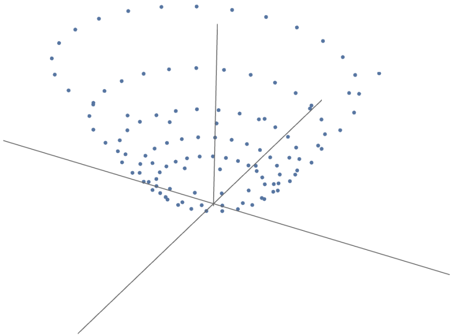
$$\Longrightarrow$$

*algebraic sets* $\equiv$ *closed sets* in the **Zariski topology**

$\overline{S}$ $=$ 'smallest algebraic set containing $S$'

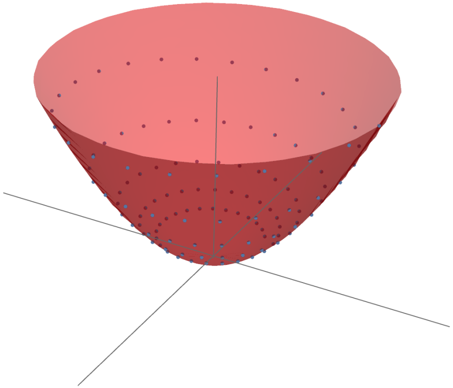
## Zariski Closure for a Single Loop

$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$



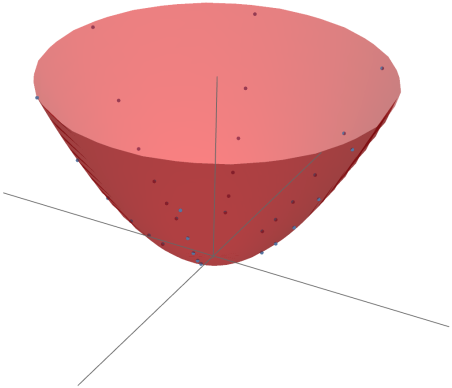
## Zariski Closure for a Single Loop

$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$



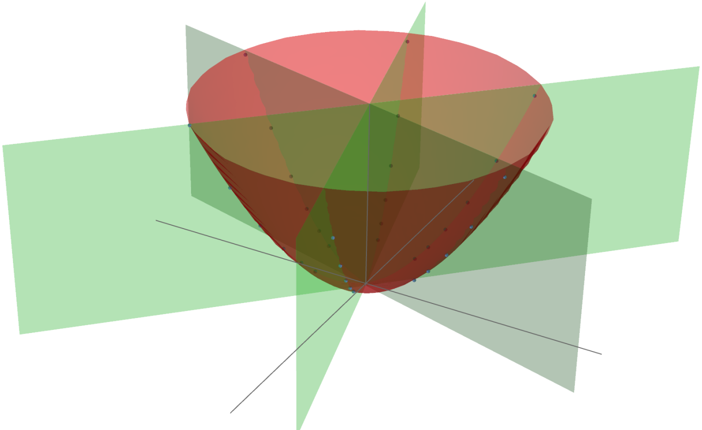
## Zariski Closure for a Single Loop

$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$



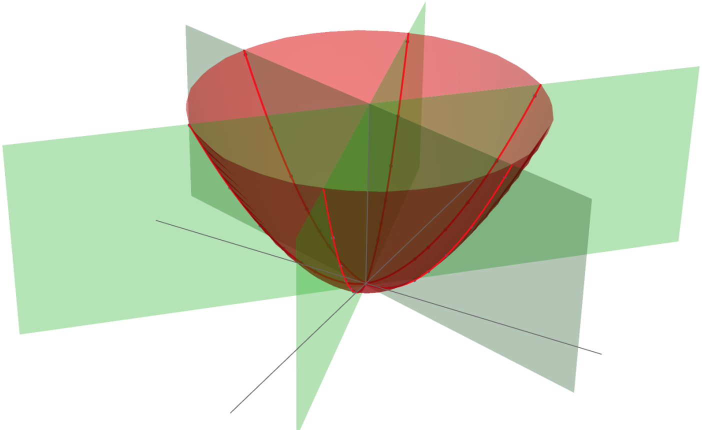
## Zariski Closure for a Single Loop

$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$



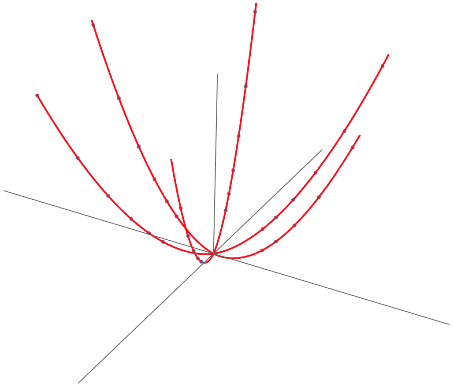
## Zariski Closure for a Single Loop

$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

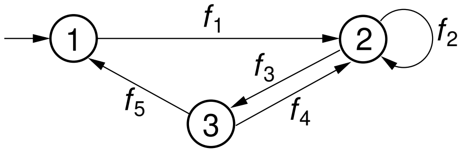


## Zariski Closure for a Single Loop

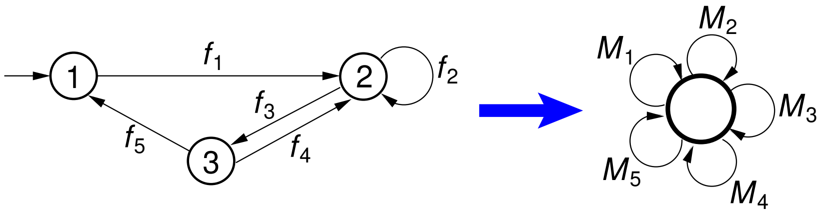
$$\begin{pmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ 0 & 0 & 4 \end{pmatrix}$$



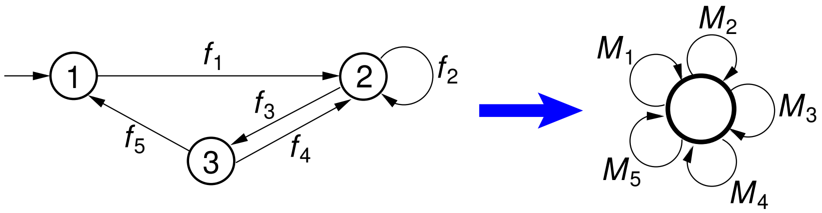
## From Affine Programs to Linear Semigroups



## From Affine Programs to Linear Semigroups



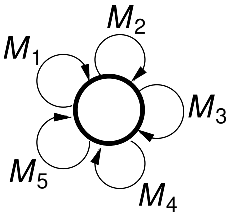
## From Affine Programs to Linear Semigroups



each $M_i \in \mathbb{Q}^{d \times d}$

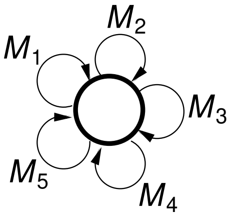
## Zariski Closure of Linear Semigroups

- $M_1, \ldots, M_k \in \mathbb{Q}^{d \times d}$



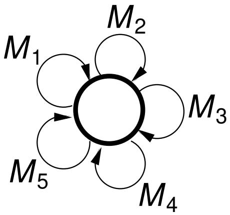
## Zariski Closure of Linear Semigroups

- $M_1, \ldots, M_k \in \mathbb{Q}^{d \times d}$
- Linear semigroup $\langle M_1, \ldots, M_k \rangle \subseteq \mathbb{Q}^{d \times d}$



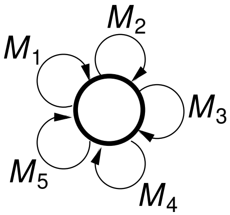
## Zariski Closure of Linear Semigroups

- $M_1, \ldots, M_k \in \mathbb{Q}^{d \times d}$
- Linear semigroup $\langle M_1, \ldots, M_k \rangle \subseteq \mathbb{Q}^{d \times d}$
- Zariski closure $\overline{\langle M_1, \ldots, M_k \rangle} \subseteq \mathbb{R}^{d \times d}$



## Zariski Closure of Linear Semigroups

- $M_1, \ldots, M_k \in \mathbb{Q}^{d \times d}$
- Linear semigroup $\langle M_1, \ldots, M_k \rangle \subseteq \mathbb{Q}^{d \times d}$
- Zariski closure $\overline{\langle M_1, \ldots, M_k \rangle} \subseteq \mathbb{R}^{d \times d}$

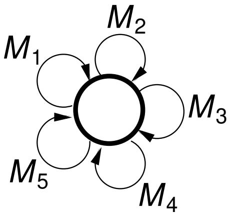


**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

*There is an algorithm which computes* $\overline{\langle M_1, \ldots, M_k \rangle}$.

## Zariski Closure of Linear Semigroups

- $M_1, \ldots, M_k \in \mathbb{Q}^{d \times d}$
- Linear semigroup $\langle M_1, \ldots, M_k \rangle \subseteq \mathbb{Q}^{d \times d}$
- Zariski closure $\overline{\langle M_1, \ldots, M_k \rangle} \subseteq \mathbb{R}^{d \times d}$

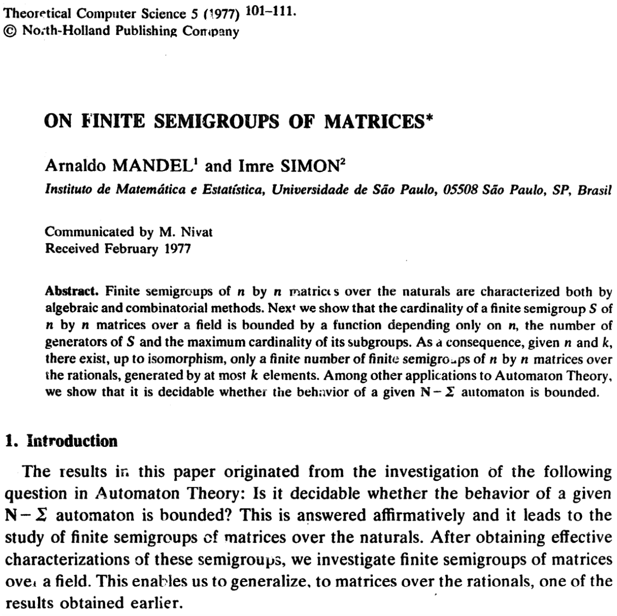


**Theorem (Hrushovski, Ouaknine, Pouly, W. 18)**

*There is an algorithm which computes* $\overline{\langle M_1, \ldots, M_k \rangle}$.

Outputs a finite list of polynomials $p_1, \ldots, p_m \in \mathbb{Z}[x_1, \ldots, x_{d^2}]$ such that:

$$\overline{\langle M_1, \ldots, M_k \rangle} = \mathbf{V}(p_1, \ldots, p_m)$$



Theoretical Computer Science 5 (1977) 101–111.
© North-Holland Publishing Company

# ON FINITE SEMIGROUPS OF MATRICES*

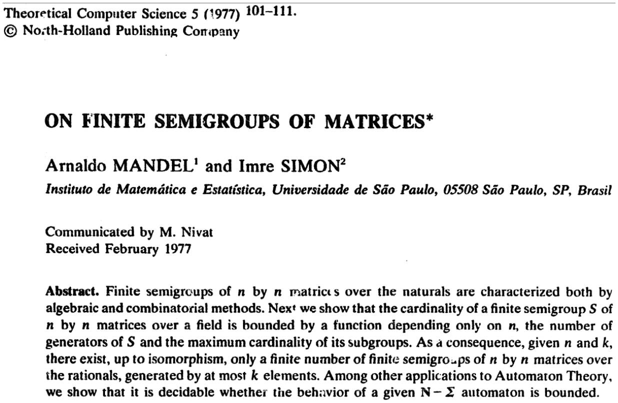
Arnaldo MANDEL[1] and Imre SIMON[2]
*Instituto de Matemática e Estatística, Universidade de São Paulo, 05508 São Paulo, SP, Brasil*

Communicated by M. Nivat
Received February 1977

**Abstract.** Finite semigroups of $n$ by $n$ matrices over the naturals are characterized both by algebraic and combinatorial methods. Next we show that the cardinality of a finite semigroup $S$ of $n$ by $n$ matrices over a field is bounded by a function depending only on $n$, the number of generators of $S$ and the maximum cardinality of its subgroups. As a consequence, given $n$ and $k$, there exist, up to isomorphism, only a finite number of finite semigroups of $n$ by $n$ matrices over the rationals, generated by at most $k$ elements. Among other applications to Automaton Theory, we show that it is decidable whether the behavior of a given $\mathbf{N}-\Sigma$ automaton is bounded.

## 1. Introduction

The results in this paper originated from the investigation of the following question in Automaton Theory: Is it decidable whether the behavior of a given $\mathbf{N}-\Sigma$ automaton is bounded? This is answered affirmatively and it leads to the study of finite semigroups of matrices over the naturals. After obtaining effective characterizations of these semigroups, we investigate finite semigroups of matrices over a field. This enables us to generalize, to matrices over the rationals, one of the results obtained earlier.



Theoretical Computer Science 5 (1977) 101–111.
© North-Holland Publishing Company

ON FINITE SEMIGROUPS OF MATRICES*

Arnaldo MANDEL[1] and Imre SIMON[2]
*Instituto de Matemática e Estatística, Universidade de São Paulo, 05508 São Paulo, SP, Brasil*

Communicated by M. Nivat
Received February 1977

**Abstract.** Finite semigroups of $n$ by $n$ matrices over the naturals are characterized both by algebraic and combinatorial methods. Next we show that the cardinality of a finite semigroup $S$ of $n$ by $n$ matrices over a field is bounded by a function depending only on $n$, the number of generators of $S$ and the maximum cardinality of its subgroups. As a consequence, given $n$ and $k$, there exist, up to isomorphism, only a finite number of finite semigroups of $n$ by $n$ matrices over the rationals, generated by at most $k$ elements. Among other applications to Automaton Theory, we show that it is decidable whether the behavior of a given $\mathbf{N}-\Sigma$ automaton is bounded.

$$\langle M_1, \ldots, M_k \rangle \text{ is finite}$$

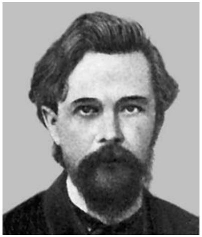
$$\Longleftrightarrow$$

$$\overline{\langle M_1, \ldots, M_k \rangle} \text{ is finite!}$$

## Some Hard Problems for Linear Semigroups

**Theorem (Markov 1947)**

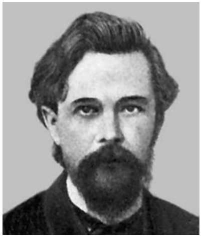
*There is a fixed set of* $6 \times 6$ *integer matrices* $M_1, \ldots, M_k$ *such that the membership problem "*$M \in \langle M_1, \ldots, M_k \rangle$*?" is* **undecidable**.



## Some Hard Problems for Linear Semigroups

**Theorem (Markov 1947)**

*There is a fixed set of $6 \times 6$ integer matrices $M_1, \ldots, M_k$ such that the membership problem "$M \in \langle M_1, \ldots, M_k \rangle$?" is* **undecidable**.

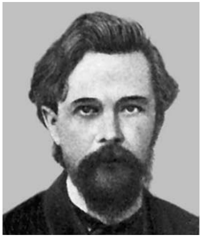


**Mortality:** Is the zero matrix contained in the semigroup generated by a given set of $n \times n$ matrices with integer entries?

## Some Hard Problems for Linear Semigroups

**Theorem (Markov 1947)**

*There is a fixed set of $6 \times 6$ integer matrices $M_1, \dots, M_k$ such that the membership problem "$M \in \langle M_1, \dots, M_k \rangle$?" is* **undecidable***.*

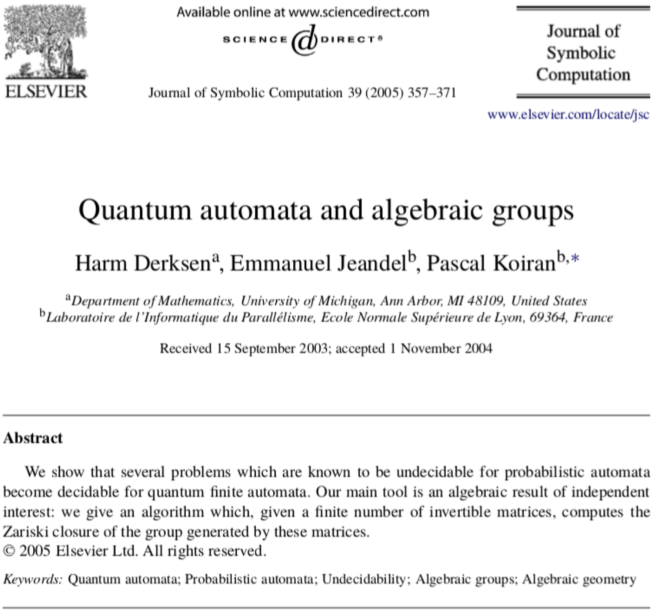


**Mortality:** Is the zero matrix contained in the semigroup generated by a given set of $n \times n$ matrices with integer entries?

**Theorem (Paterson 1970)**

*The mortality problem is* **undecidable** *for $3 \times 3$ matrices.*

## The Group Case



Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Symbolic Computation 39 (2005) 357–371

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Quantum automata and algebraic groups

Harm Derksen[a], Emmanuel Jeandel[b], Pascal Koiran[b,*]

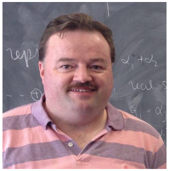
[a]*Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, United States*
[b]*Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 69364, France*

Received 15 September 2003; accepted 1 November 2004

**Abstract**

We show that several problems which are known to be undecidable for probabilistic automata become decidable for quantum finite automata. Our main tool is an algebraic result of independent interest: we give an algorithm which, given a finite number of invertible matrices, computes the Zariski closure of the group generated by these matrices.

*Keywords:* Quantum automata; Probabilistic automata; Undecidability; Algebraic groups; Algebraic geometry

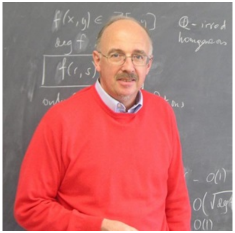


## Ingredients

**Theorem (Masser 1988)**

*Given algebraic numbers $\lambda_1, \ldots, \lambda_k$, there is a procedure to compute the set of* **multiplicative relations**

$$\{(n_1, \ldots, n_k) \in \mathbb{Z}^k : \lambda_1^{n_1} \cdots \lambda_k^{n_k} = 1\}.$$

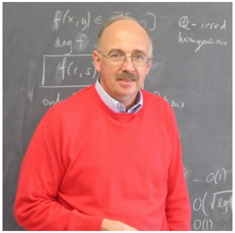


## Ingredients

**Theorem (Masser 1988)**

*Given algebraic numbers $\lambda_1, \ldots, \lambda_k$, there is a procedure to compute the set of* **multiplicative relations**

$$\{(n_1, \ldots, n_k) \in \mathbb{Z}^k : \lambda_1^{n_1} \cdots \lambda_k^{n_k} = 1\}.$$



**Theorem (Schur 1911)**

*Every finitely generated periodic subgroup of* $\mathrm{GL}_n(\mathbb{C})$ *is finite.*

## From Groups to Semigroups

Define $\boldsymbol{G} := \overline{\langle S, T, E \rangle}$, where

$$S := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad E := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

## From Groups to Semigroups

Define $\boldsymbol{G} := \overline{\langle S, T, E \rangle}$, where

$$S := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad E := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Then $\boldsymbol{G} = \{M \in \mathrm{M}_2(\mathbb{R}) : \det(M) \in \{0, 1\}\}$.

## From Groups to Semigroups

Define $\boldsymbol{G} := \overline{\langle S, T, E \rangle}$, where

$$S := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad E := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Then $\boldsymbol{G} = \{M \in \mathrm{M}_2(\mathbb{R}) : \det(M) \in \{0, 1\}\}$.

Indeed, since

$$\{M \in \boldsymbol{G} : \mathrm{rank}(M) = 2\} = \overline{\langle S, T \rangle} = \overline{\mathrm{SL}_2(\mathbb{Z})} = \mathrm{SL}_2(\mathbb{R}),$$

## From Groups to Semigroups

Define $\boldsymbol{G} := \overline{\langle S, T, E \rangle}$, where

$$S := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad E := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Then $\boldsymbol{G} = \{M \in \mathrm{M}_2(\mathbb{R}) : \det(M) \in \{0, 1\}\}$.

Indeed, since

$$\{M \in \boldsymbol{G} : \mathrm{rank}(M) = 2\} = \overline{\langle S, T \rangle} = \overline{\mathrm{SL}_2(\mathbb{Z})} = \mathrm{SL}_2(\mathbb{R}),$$

we have that $\{M \in \boldsymbol{G} : \mathrm{rank}(M) < 2\}$ is generated by the **definable set**

$$\{MEM', ME, EM : M, M' \in \mathrm{SL}_2(\mathbb{R})\}.$$

## Main Technical Lemma

**Lemma**

*Given a definable set $\boldsymbol{A} \subseteq M_n(\mathbb{C})$ of rank-$r$ matrices, there is a procedure to compute*

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \mathrm{rank}(A) = r\}}\,.$$

## Structure in Algebraic Semigroups

Want to compute

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \mathrm{rank}(A) = r\}}\,.$$

## Structure in Algebraic Semigroups

Want to compute

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \mathrm{rank}(A) = r\}}\,.$$

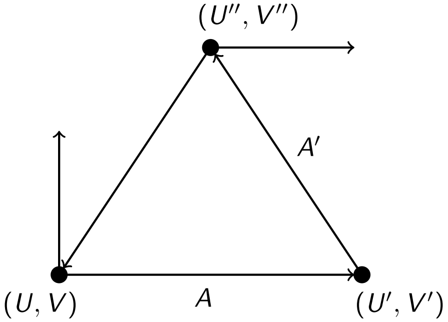
Consider $\boldsymbol{S}$ as a category $\mathcal{C}(\boldsymbol{S})$, rather than a semigroup.

## Structure in Algebraic Semigroups

Want to compute

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \operatorname{rank}(A) = r\}}\,.$$

Consider $\boldsymbol{S}$ as a category $\mathcal{C}(\boldsymbol{S})$, rather than a semigroup.

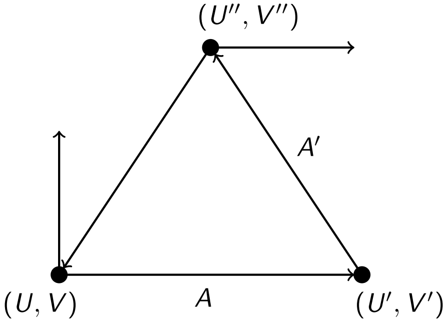


## Structure in Algebraic Semigroups

Want to compute

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \operatorname{rank}(A) = r\}}\,.$$

Consider $\boldsymbol{S}$ as a category $\mathcal{C}(\boldsymbol{S})$, rather than a semigroup.



Object $(U, V)$ s.t. $U, V \subseteq \mathbb{C}^n$

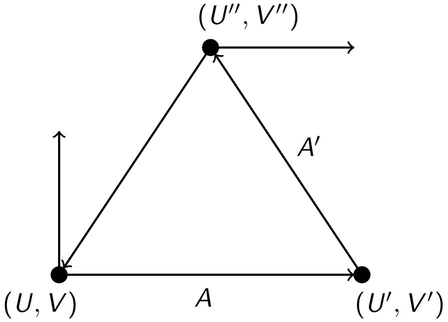
- $U \cap V = 0$
- $\dim(U) = n - r$, $\dim(V) = r$

## Structure in Algebraic Semigroups

Want to compute

$$\boldsymbol{S} = \overline{\{A \in \langle \boldsymbol{A} \rangle : \mathrm{rank}(A) = r\}}\,.$$

Consider $\boldsymbol{S}$ as a category $\mathcal{C}(\boldsymbol{S})$, rather than a semigroup.



Object $(U, V)$ s.t. $U, V \subseteq \mathbb{C}^n$

- $U \cap V = 0$
- $\dim(U) = n - r$, $\dim(V) = r$

Arrow $(U, V) \to (U', V')$:

- $A \in \boldsymbol{S}$ s.t. $\ker(A) = U$, $\mathrm{Im}(A) = V'$

## Properties of $\mathcal{C}(\boldsymbol{S})$:

- Each non-trivial SCC is a groupoid.

## Properties of $\mathcal{C}(\boldsymbol{S})$:

- Each non-trivial SCC is a groupoid.
- The number of non-trivial SCCs is at most $\binom{n}{r}$.

## Properties of $\mathcal{C}(\boldsymbol{S})$:

- Each non-trivial SCC is a groupoid.
- The number of non-trivial SCCs is at most $\binom{n}{r}$.
- The depth of the dag of SCCs is at most $2\binom{n}{r}^2$.

## Properties of $\mathcal{C}(\boldsymbol{S})$:

- Each non-trivial SCC is a groupoid.
- The number of non-trivial SCCs is at most $\binom{n}{r}$.
- The depth of the dag of SCCs is at most $2\binom{n}{r}^2$.
- Every two objects are connected by a path of arrows from $\boldsymbol{A}$ of length at most $\binom{n}{r} + 1$.

## Properties of $\mathcal{C}(\boldsymbol{S})$:

- Each non-trivial SCC is a groupoid.
- The number of non-trivial SCCs is at most $\binom{n}{r}$.
- The depth of the dag of SCCs is at most $2\binom{n}{r}^2$.
- Every two objects are connected by a path of arrows from $\boldsymbol{A}$ of length at most $\binom{n}{r} + 1$.

To obtain $\boldsymbol{S}$ we generalise the algorithm Derken, Jeandel, and Koiran from finitely generated groups to constructibly generated groupoids.

## *Polynomial Invariants for Affine Programs*, LICS 2018

**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.*