Part III: Loop Invariants

Programming in the Jurassic Era

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

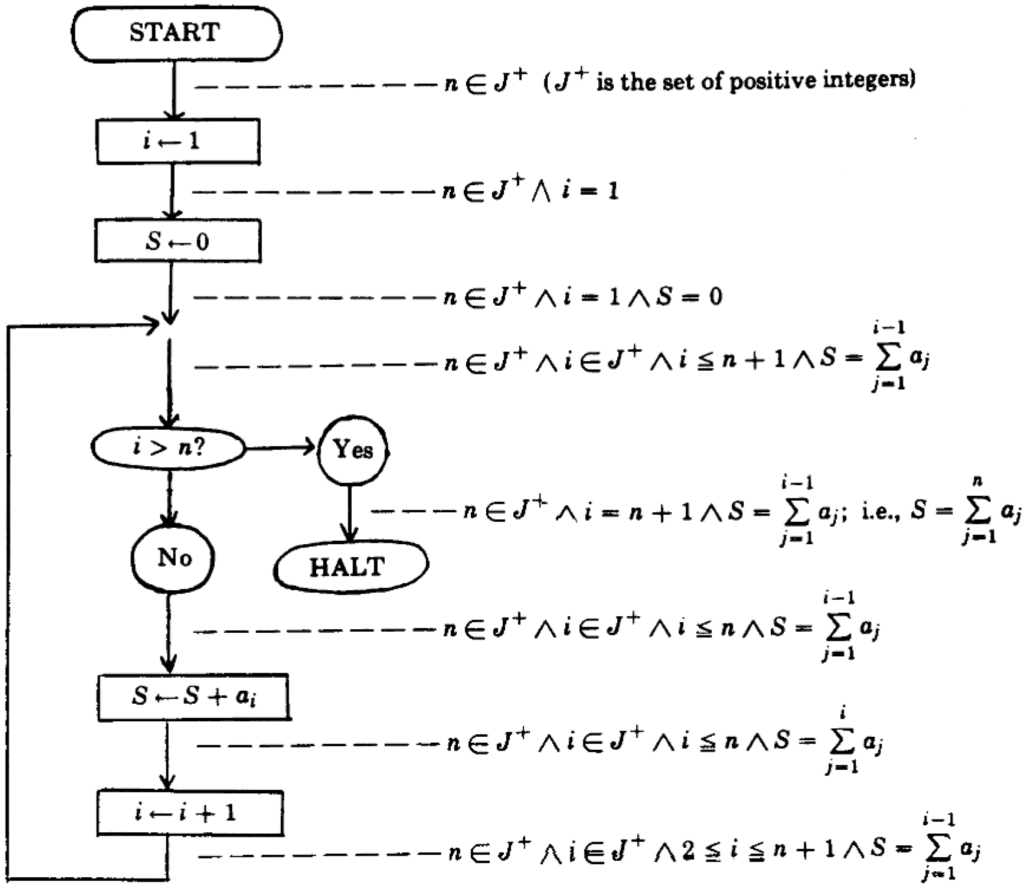
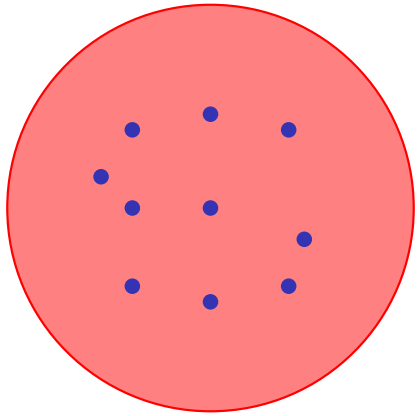


FIGURE 1. Flowchart of program to compute $S = \sum_{j=1}^{n} a_j$ $(n \geqq 0)$

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

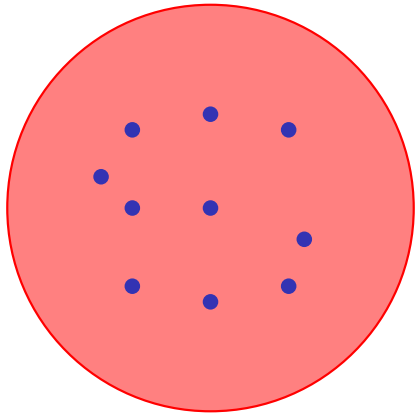
Invariants

invariant = overapproximation of the reachable states

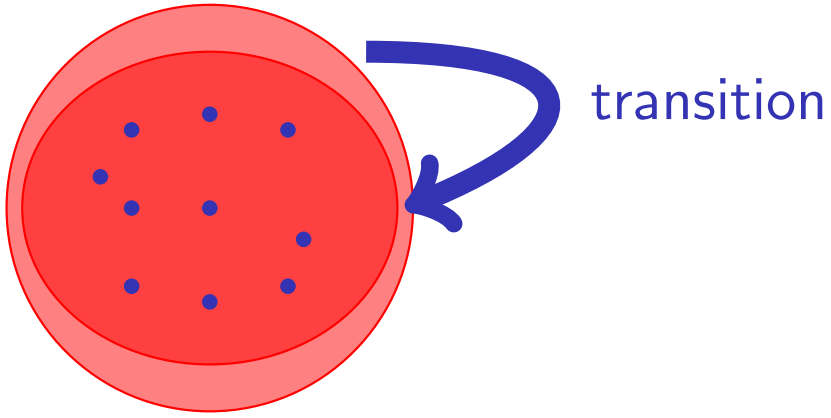


Invariants

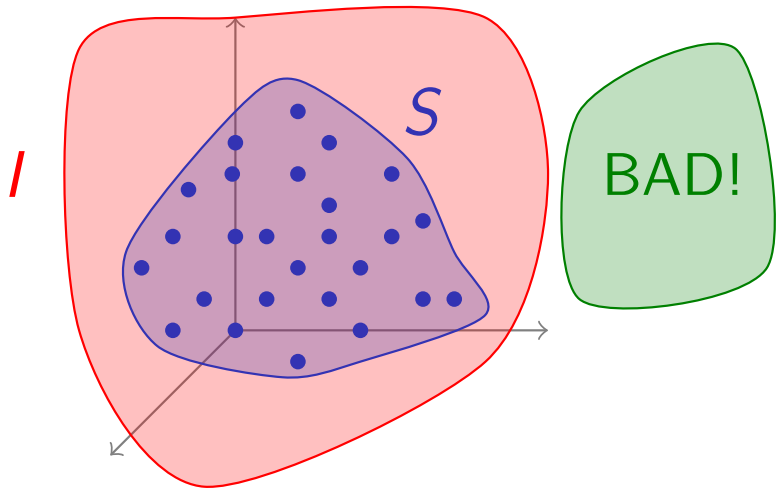
invariant = overapproximation of the reachable states



inductive invariant = invariant preserved by the transition relation



Invariant Synthesis



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

Does This Program Halt?

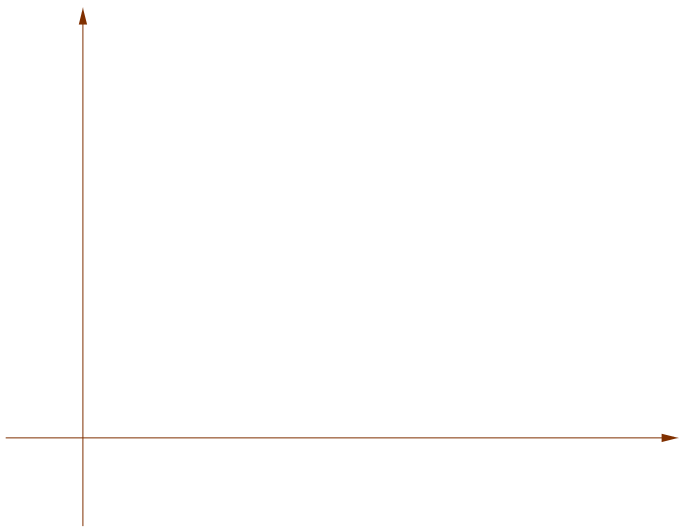
$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do

$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$

Does This Program Halt?

$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do

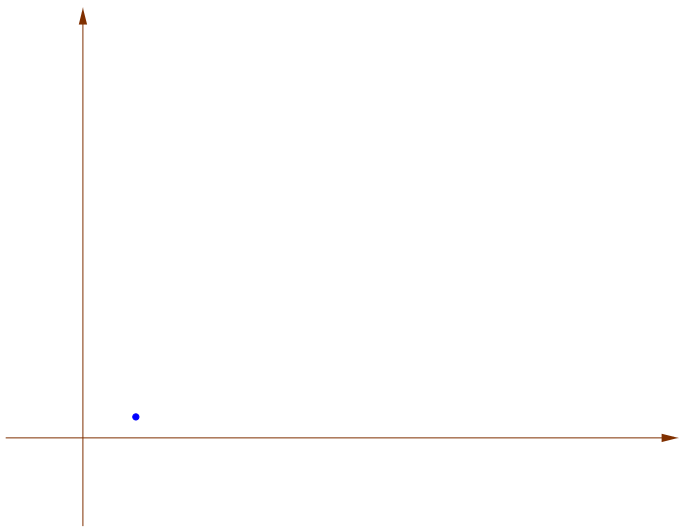
$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$



Does This Program Halt?

$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do

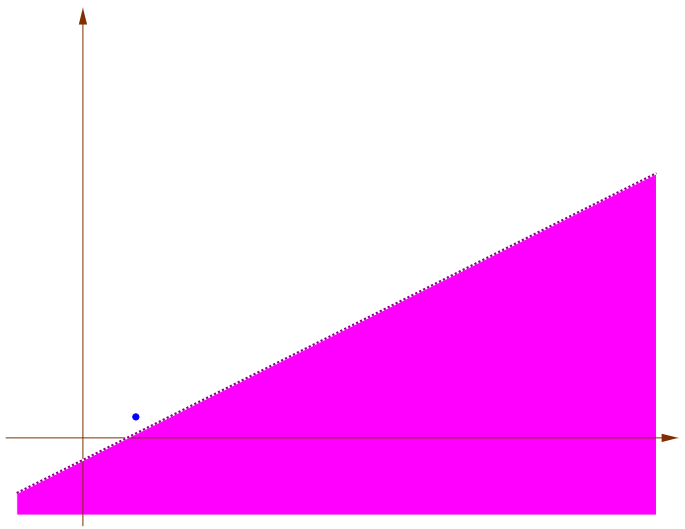
$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$



Does This Program Halt?

$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do

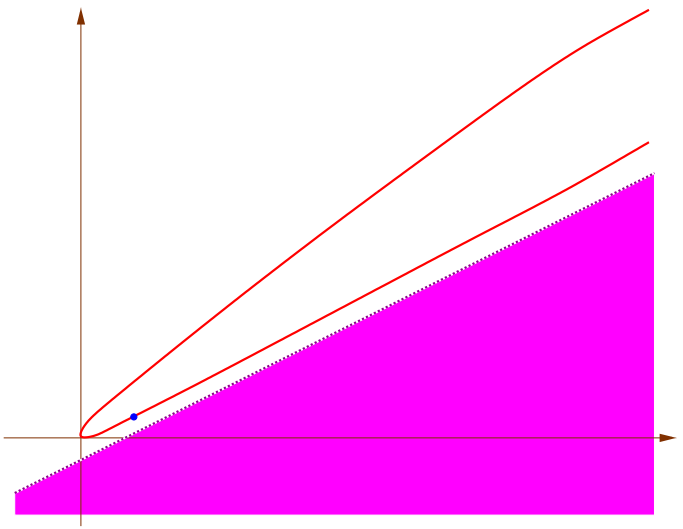
$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$



Does This Program Halt?

$x := 3;$
$y := 2;$
while $2y - x \geq -2$ do

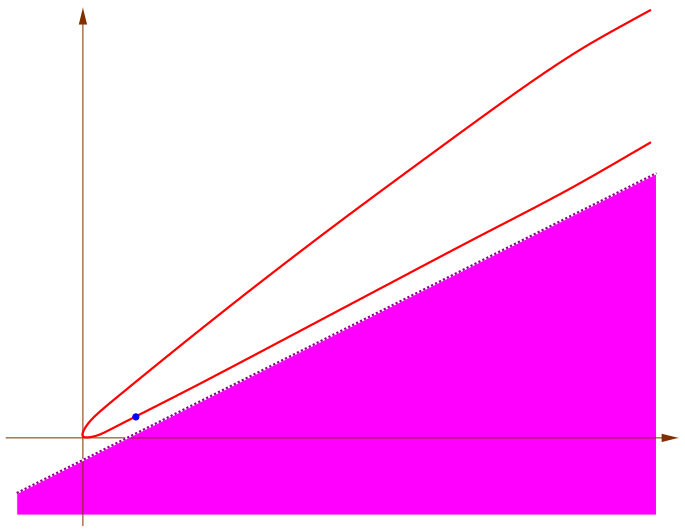
$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$



Does This Program Halt?

$x := 3;$
$y := 2;$
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: $x - 9x^2 - y + 24xy - 16y^2 = 0$

Automata-Theoretic Application of Polynomial Invariants

DECIDABLE AND UNDECIDABLE PROBLEMS ABOUT QUANTUM AUTOMATA*

VINCENT D. BLONDEL†, EMMANUEL JEANDEL‡, PASCAL KOIRAN‡, AND NATACHA PORTIER‡

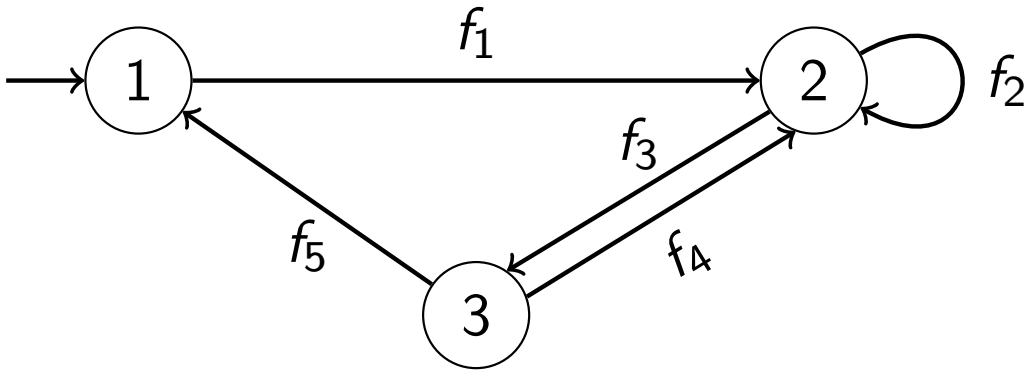
Abstract. We study the following decision problem: is the language recognized by a quantum finite automaton empty or nonempty? We prove that this problem is decidable or undecidable depending on whether recognition is defined by strict or nonstrict thresholds. This result is in contrast with the corresponding situation for probabilistic finite automata, for which it is known that strict and nonstrict thresholds both lead to undecidable problems.

Theorem (Blondel, Jeandel, Koiran, Portier 2005)

The strict threshold problem is decidable for quantum automata.

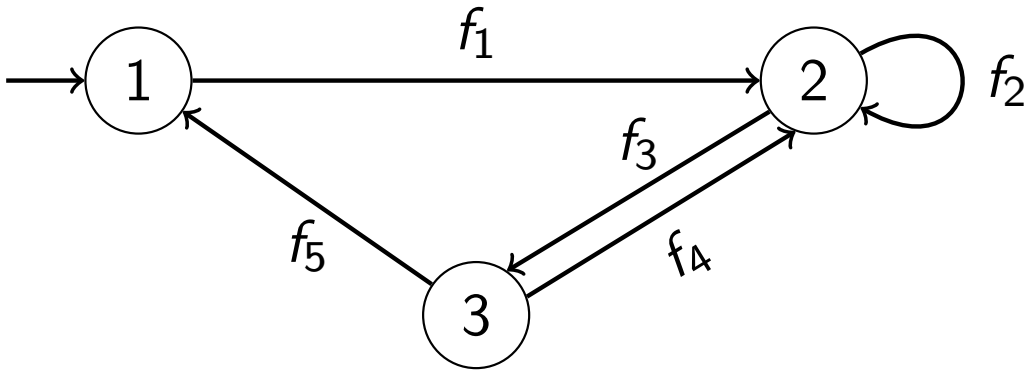
From Flowcharts to Affine Programs

Affine programs (Muller-Olm and Seidl 2004)



From Flowcharts to Affine Programs

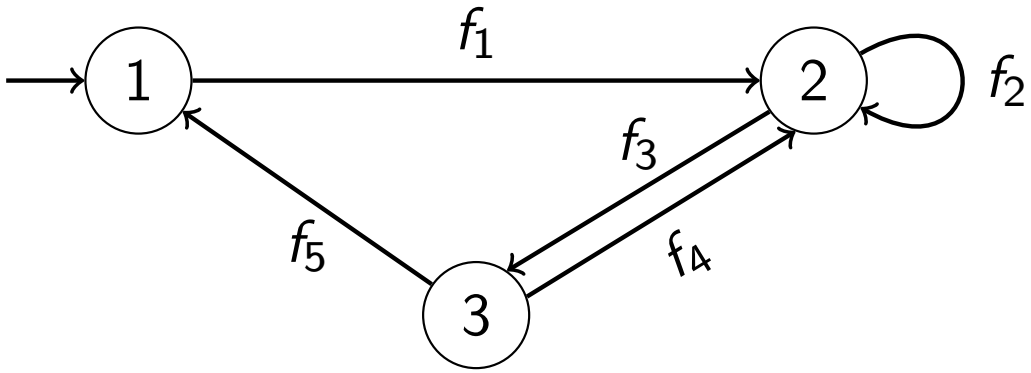
Affine programs (Muller-Olm and Seidl 2004)



- Nondeterministic branching (no guards)

From Flowcharts to Affine Programs

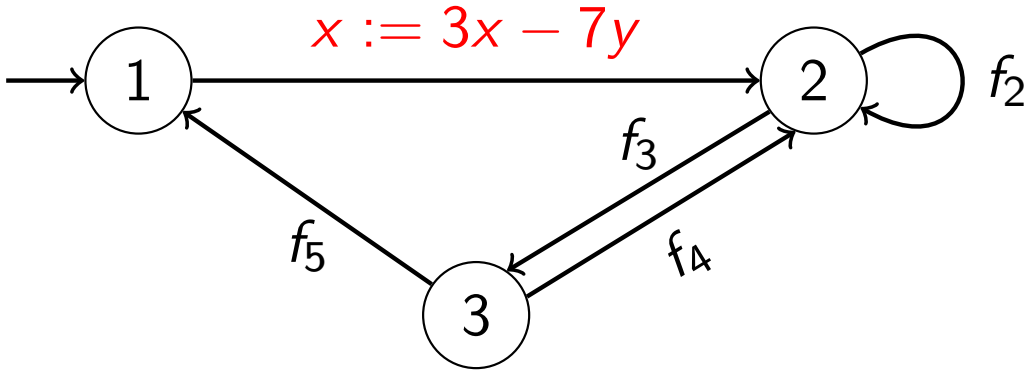
Affine programs (Muller-Olm and Seidl 2004)



- Nondeterministic branching (no guards)
- Integer variables with affine assignments

From Flowcharts to Affine Programs

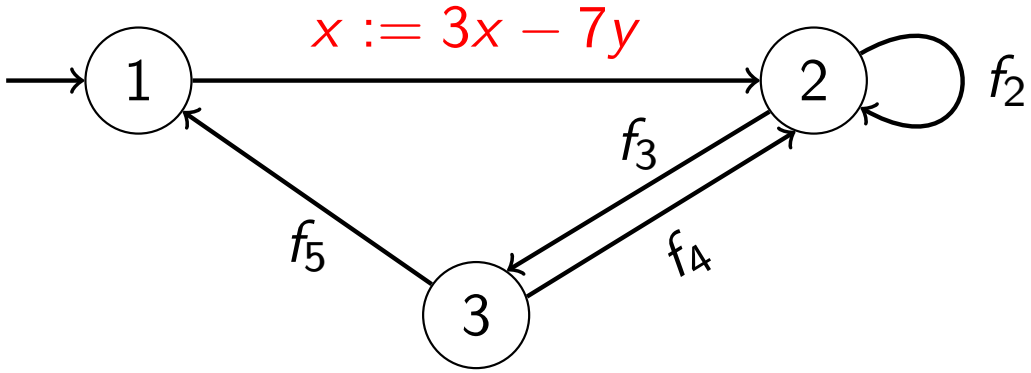
Affine programs (Muller-Olm and Seidl 2004)



- Nondeterministic branching (no guards)
- Integer variables with affine assignments

From Flowcharts to Affine Programs

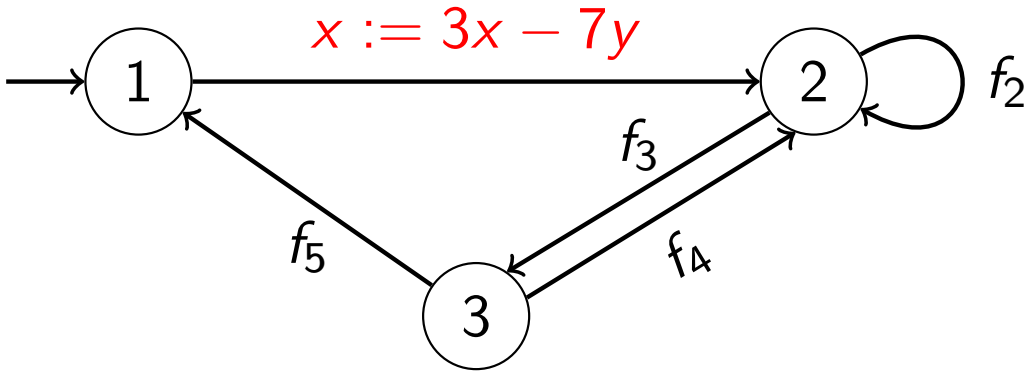
Affine programs (Muller-Olm and Seidl 2004)



- Nondeterministic branching (no guards)
- Integer variables with affine assignments
- Compute **all valid polynomial relations** at each location

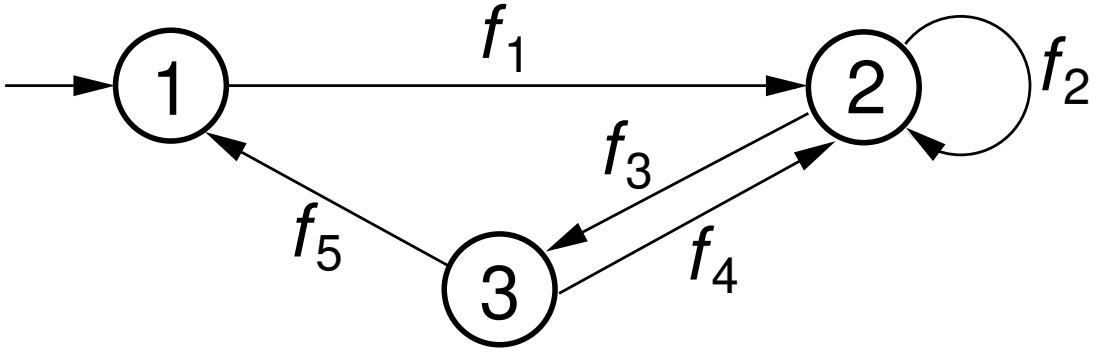
From Flowcharts to Affine Programs

Affine programs (Muller-Olm and Seidl 2004)



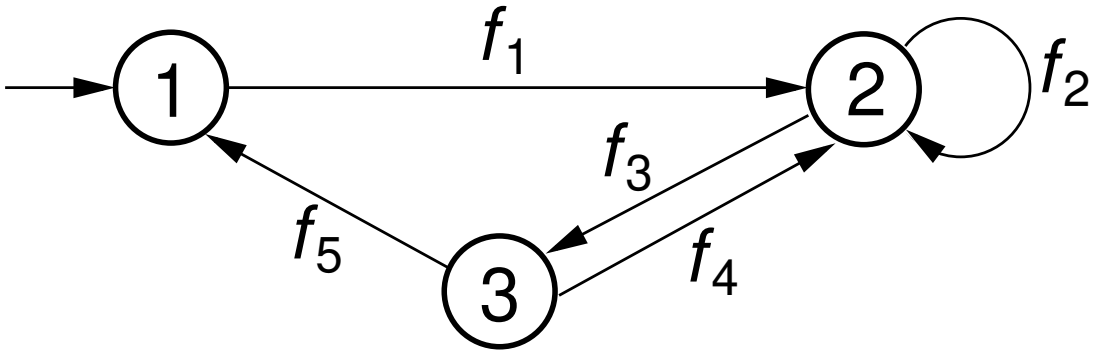
- Nondeterministic branching (no guards)
- Integer variables with affine assignments
- Compute **all valid polynomial relations** at each location
- Represents the **Zariski closure** of the reachable set at each location

Geometric Picture



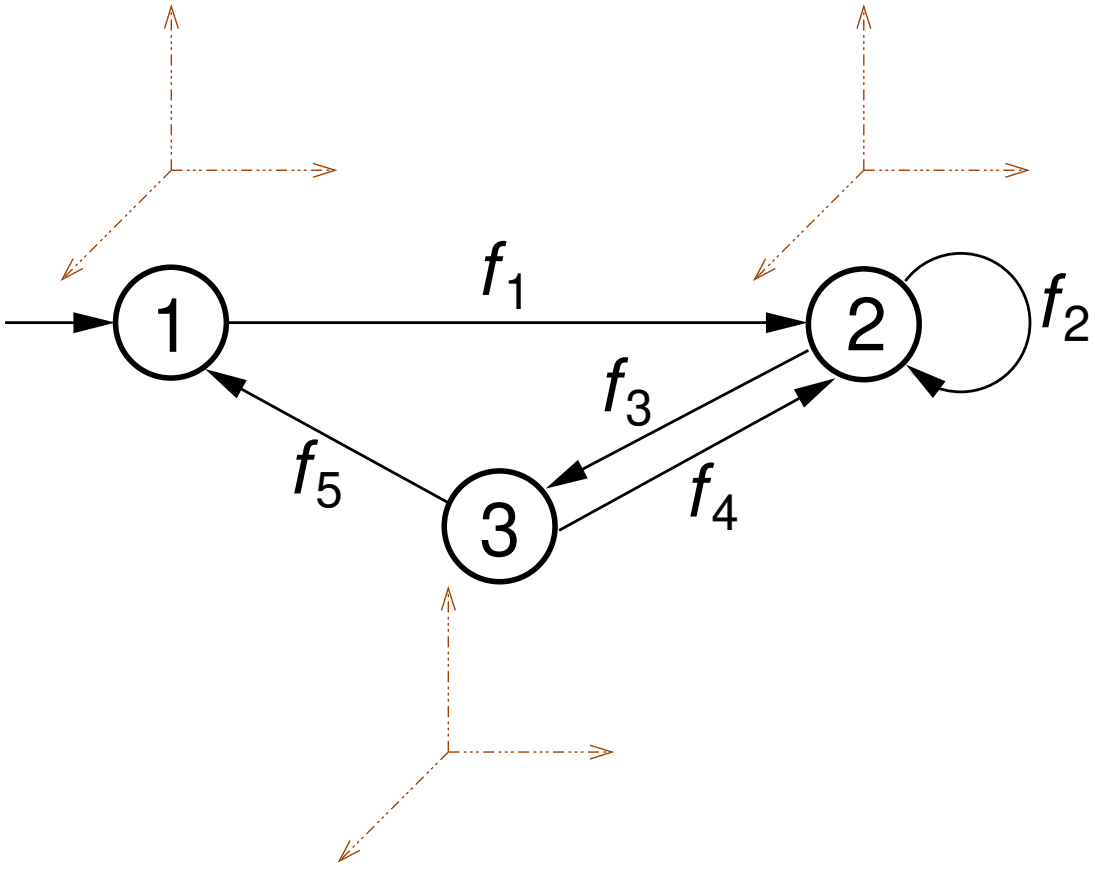
Geometric Picture

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



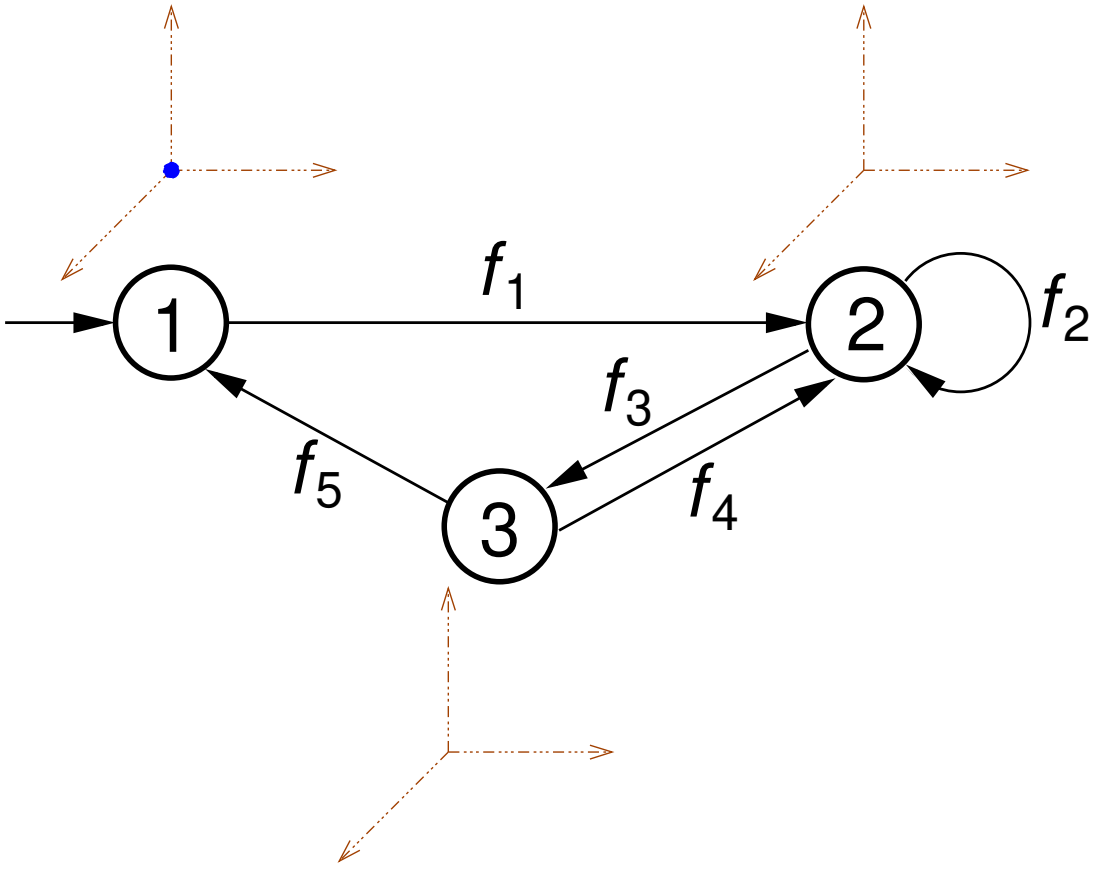
Geometric Picture

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



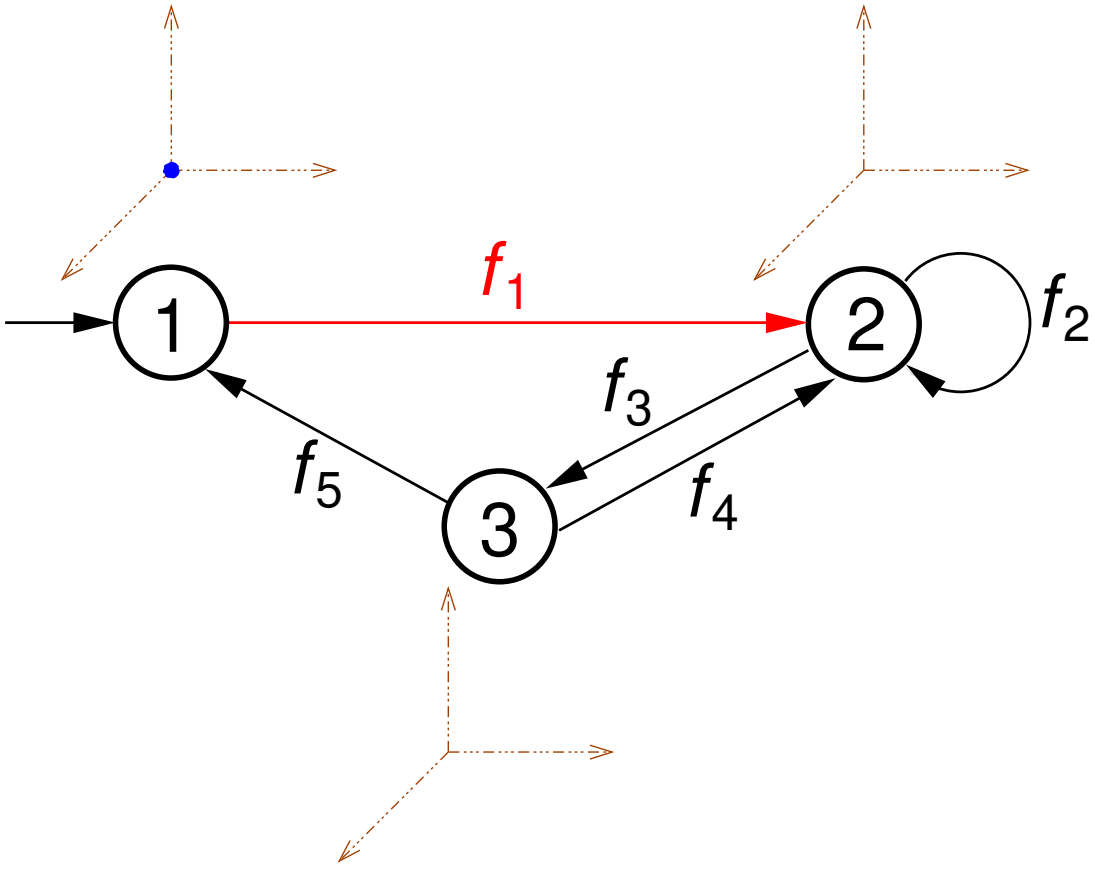
Geometric Picture

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



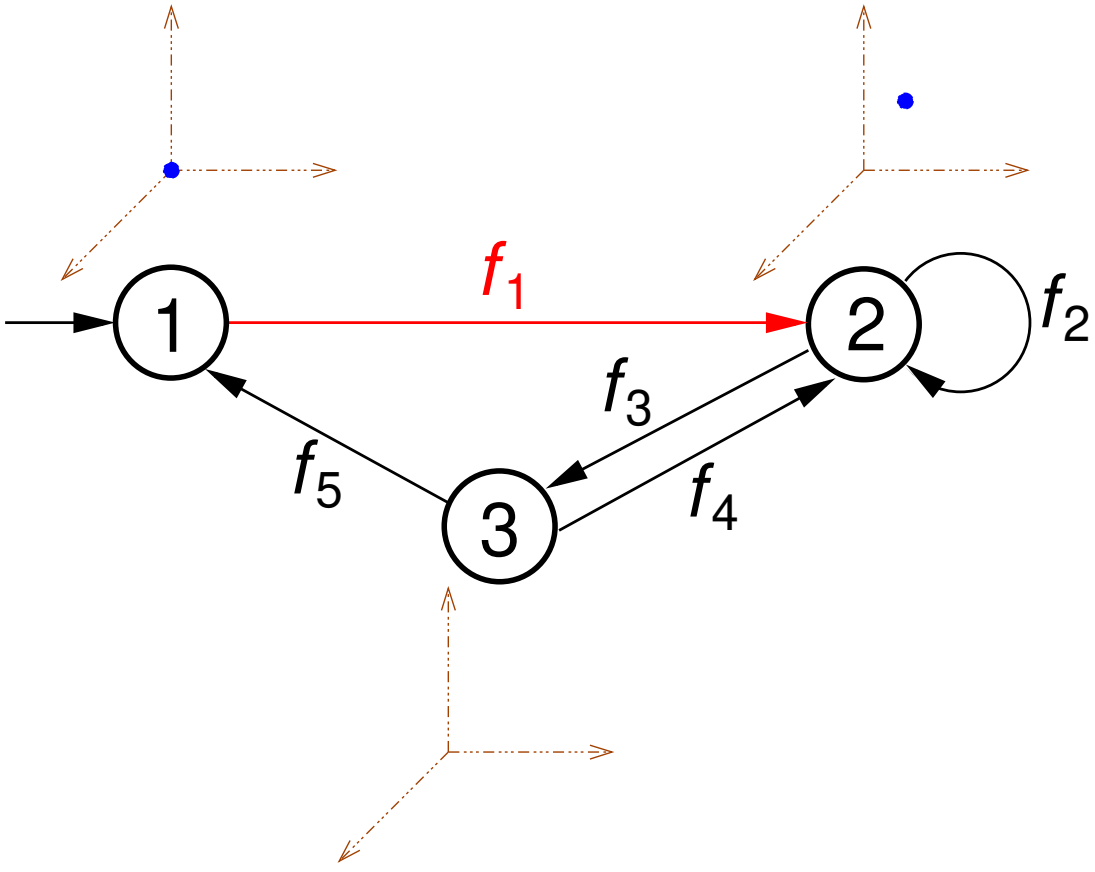
Geometric Picture

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



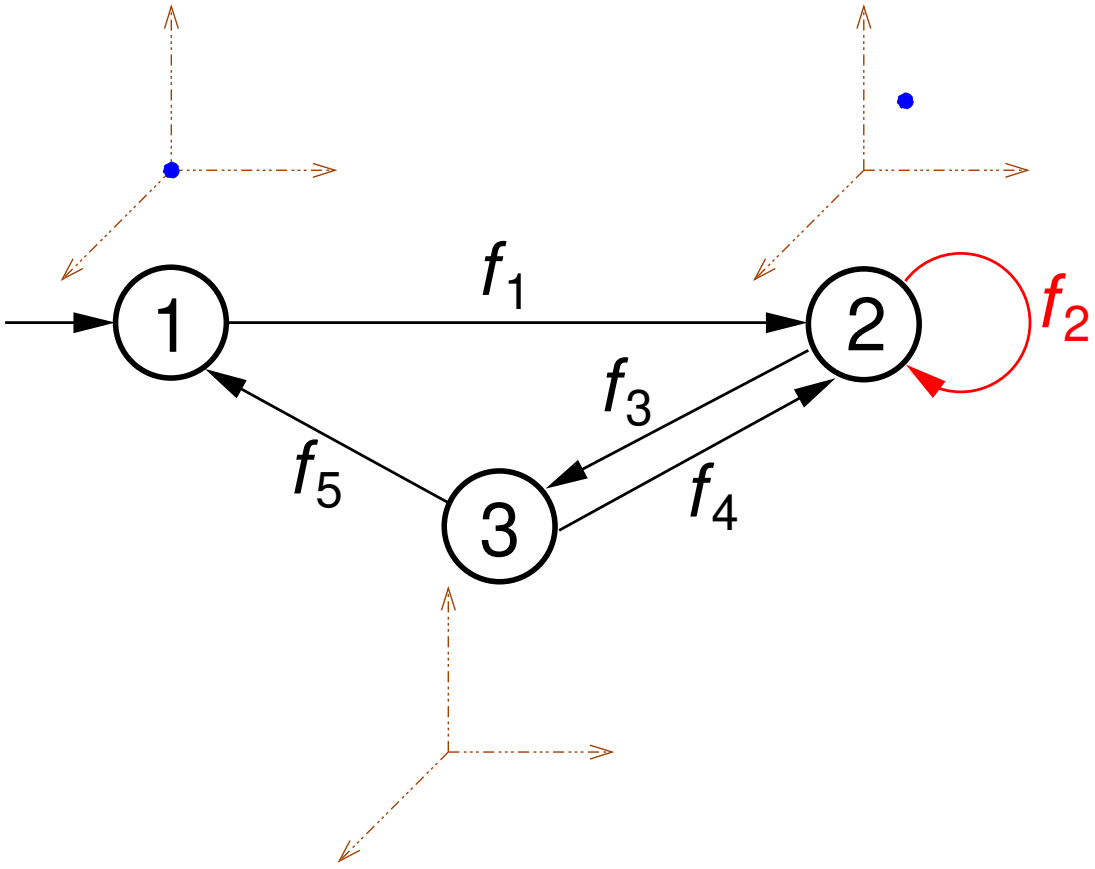
Geometric Picture

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



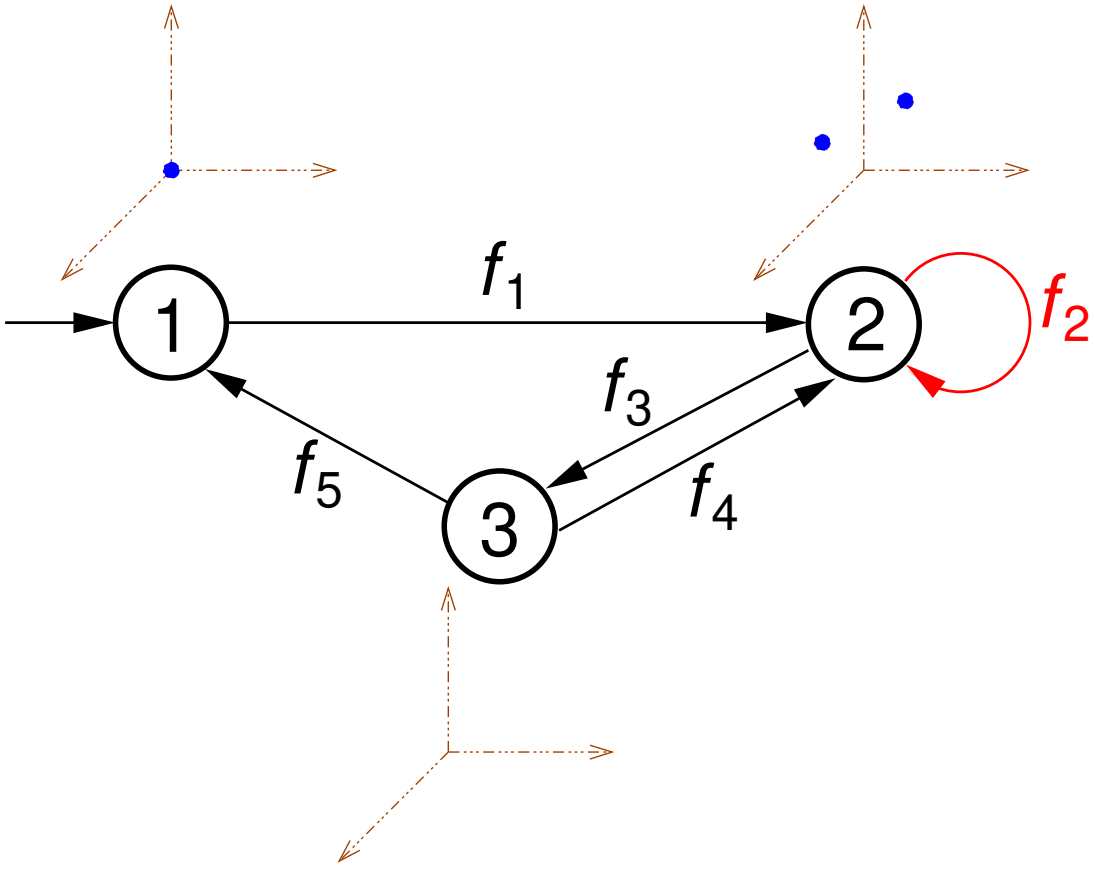
Geometric Picture

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



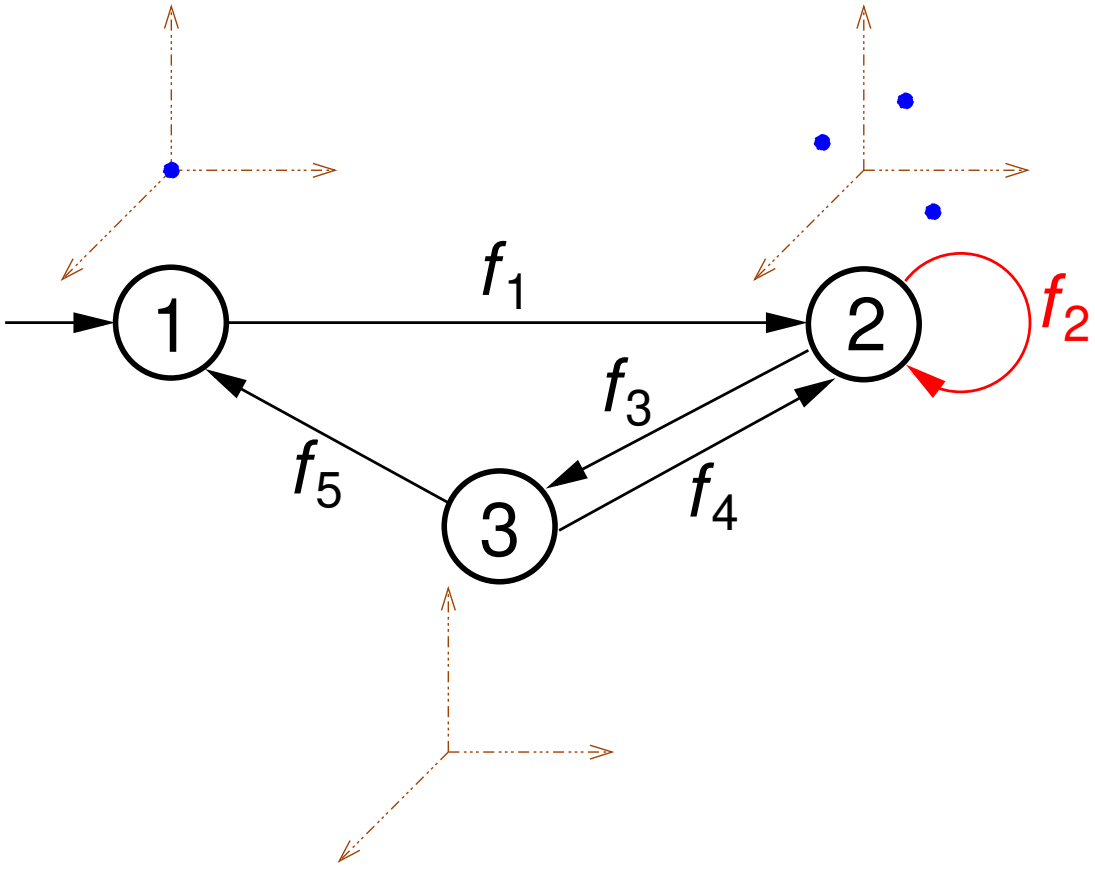
Geometric Picture

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



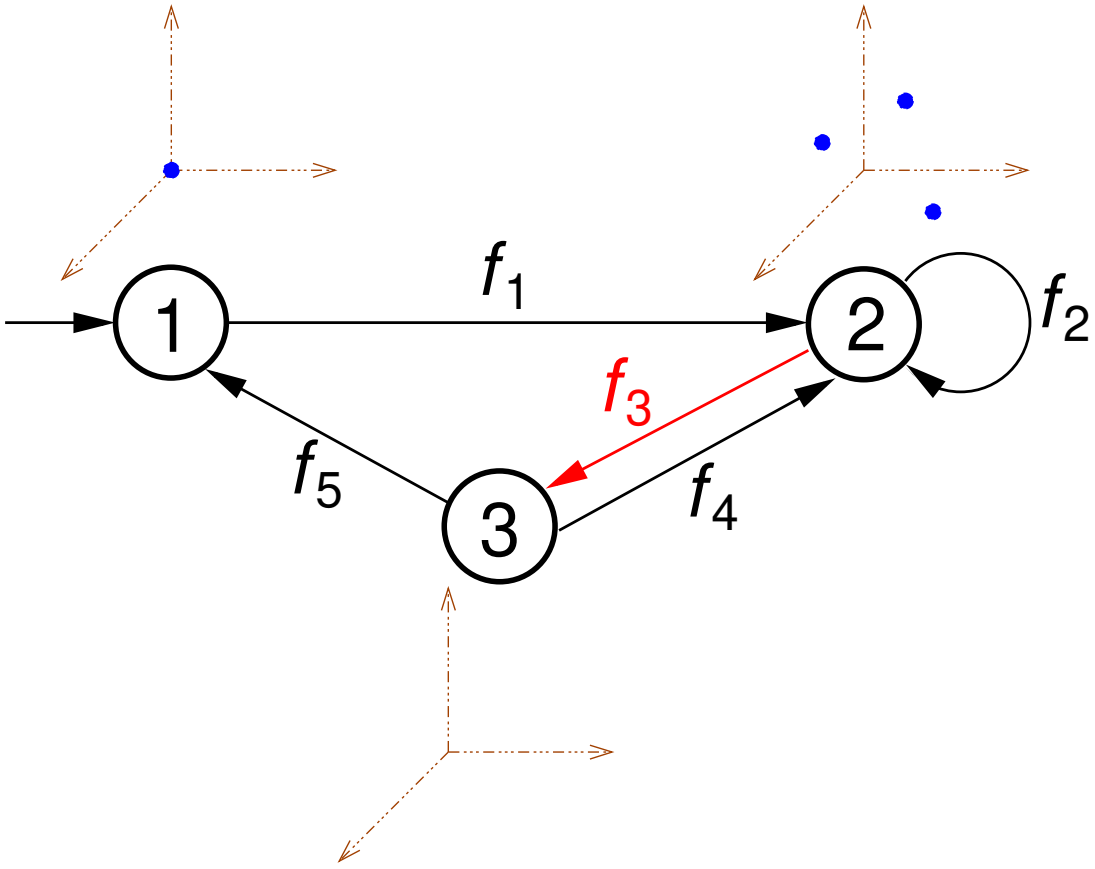
Geometric Picture

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



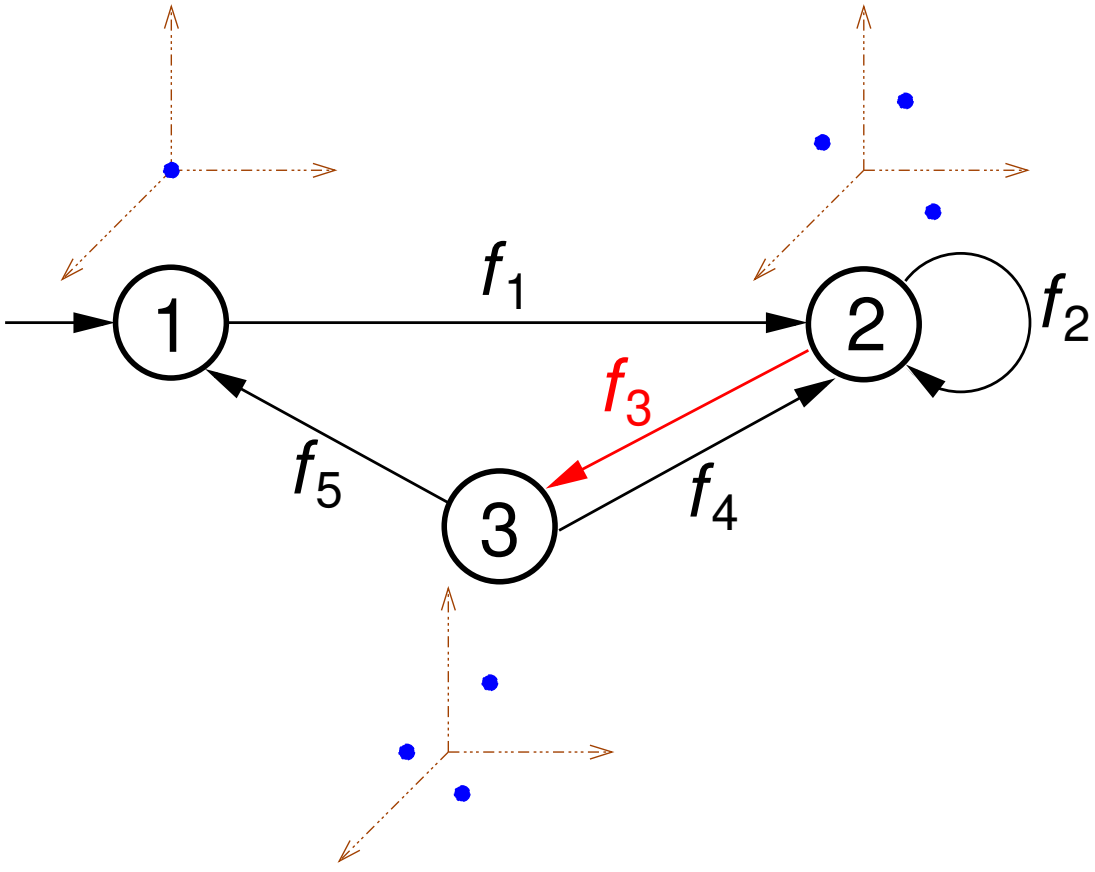
Geometric Picture

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



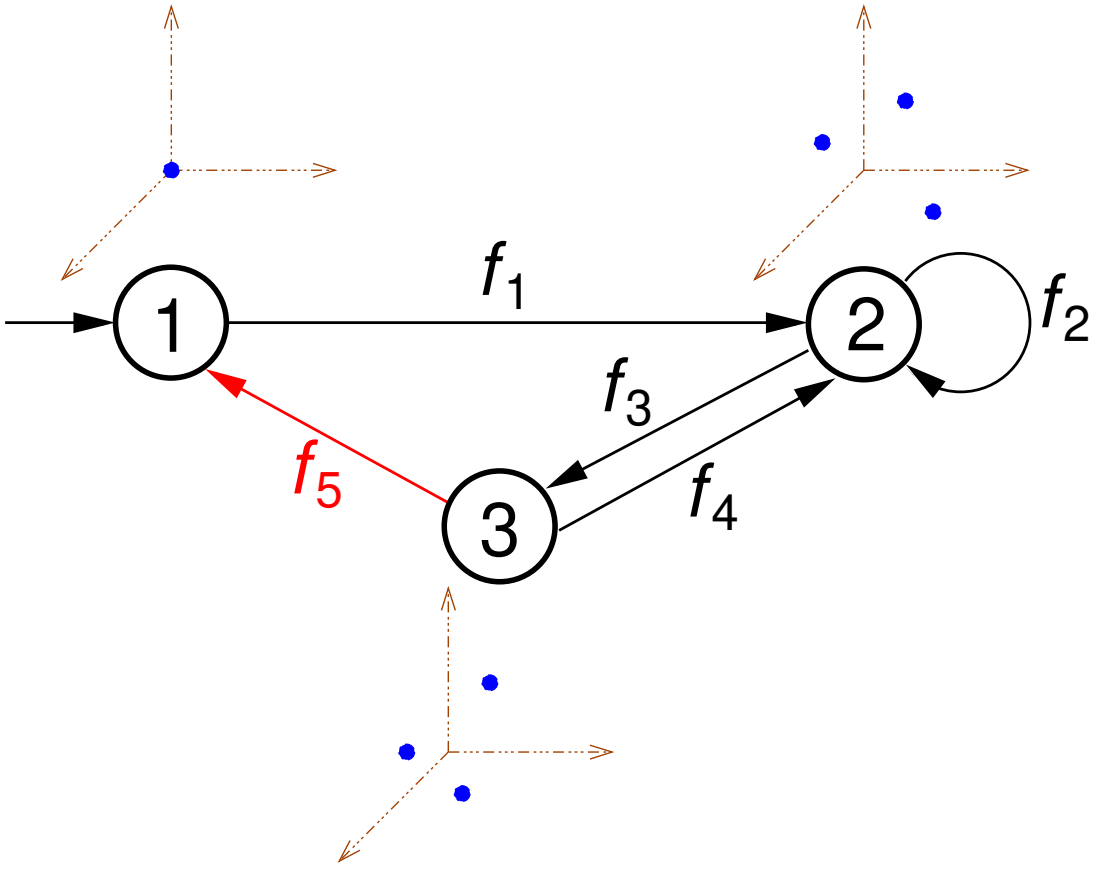
Geometric Picture

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



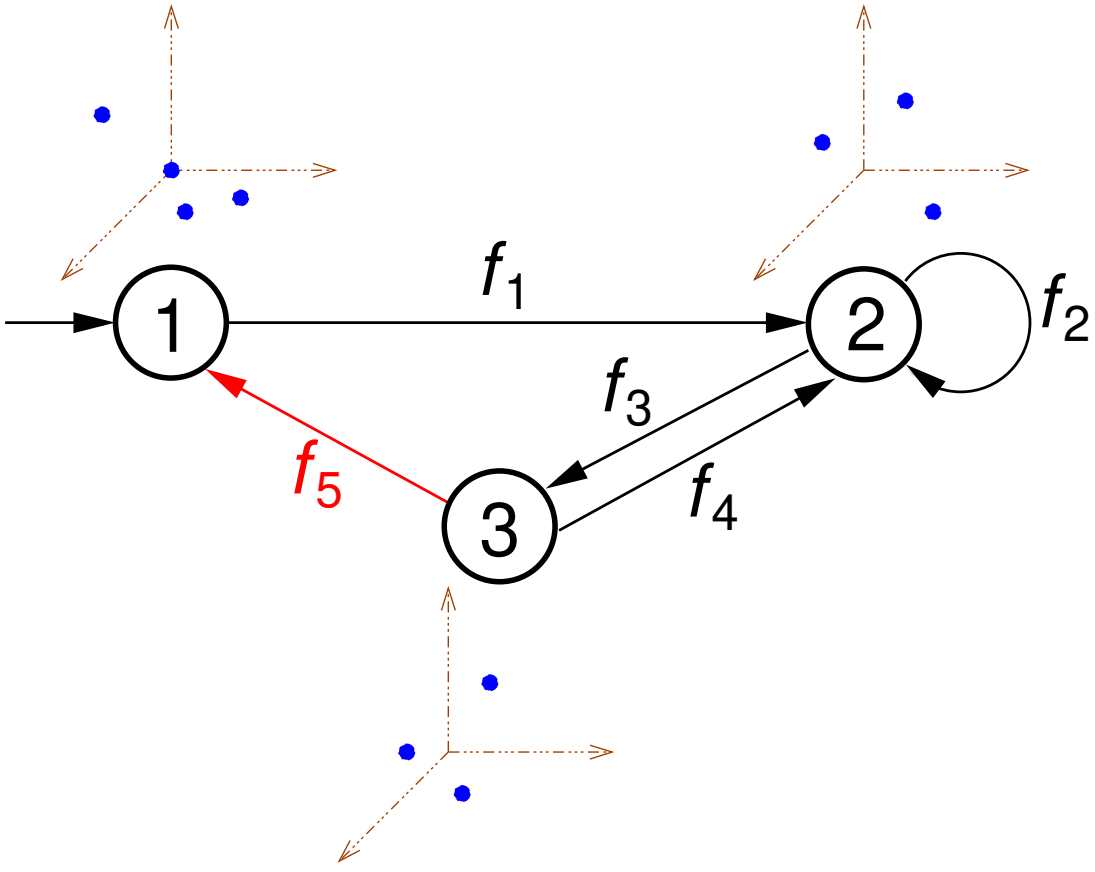
Geometric Picture

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



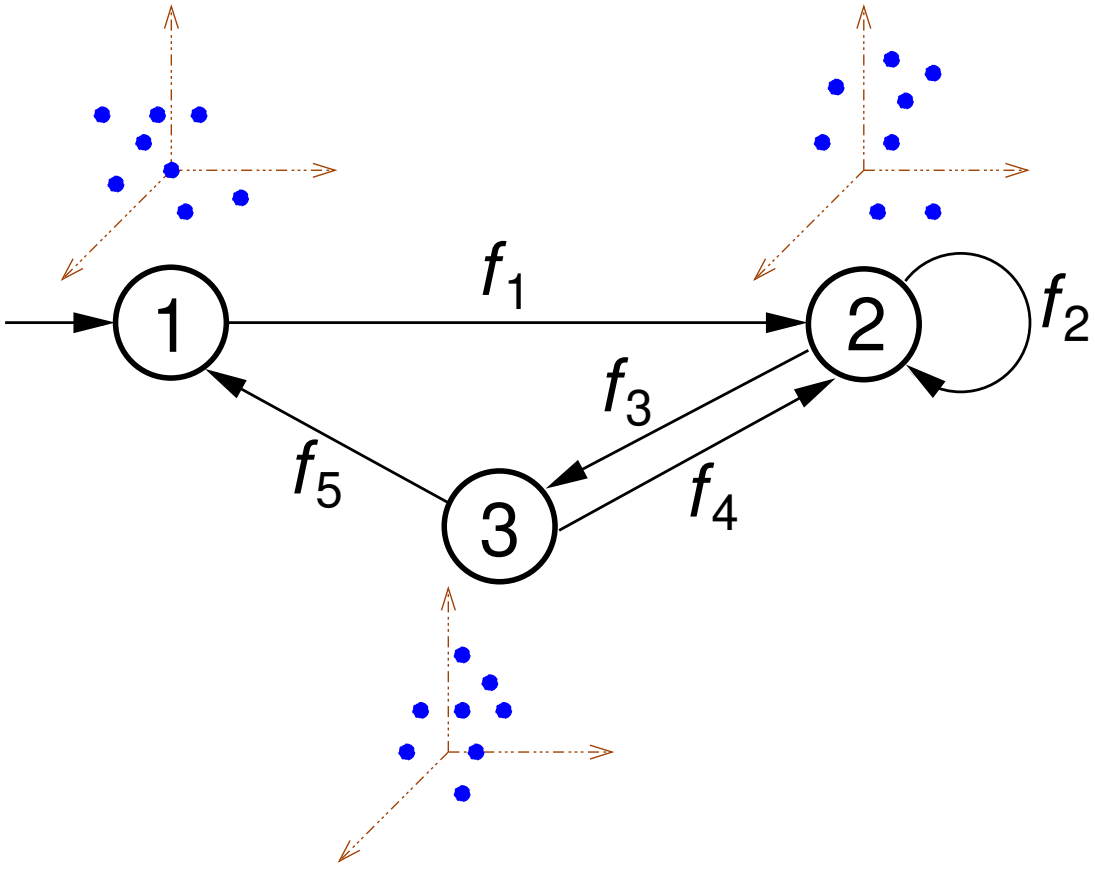
Geometric Picture

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



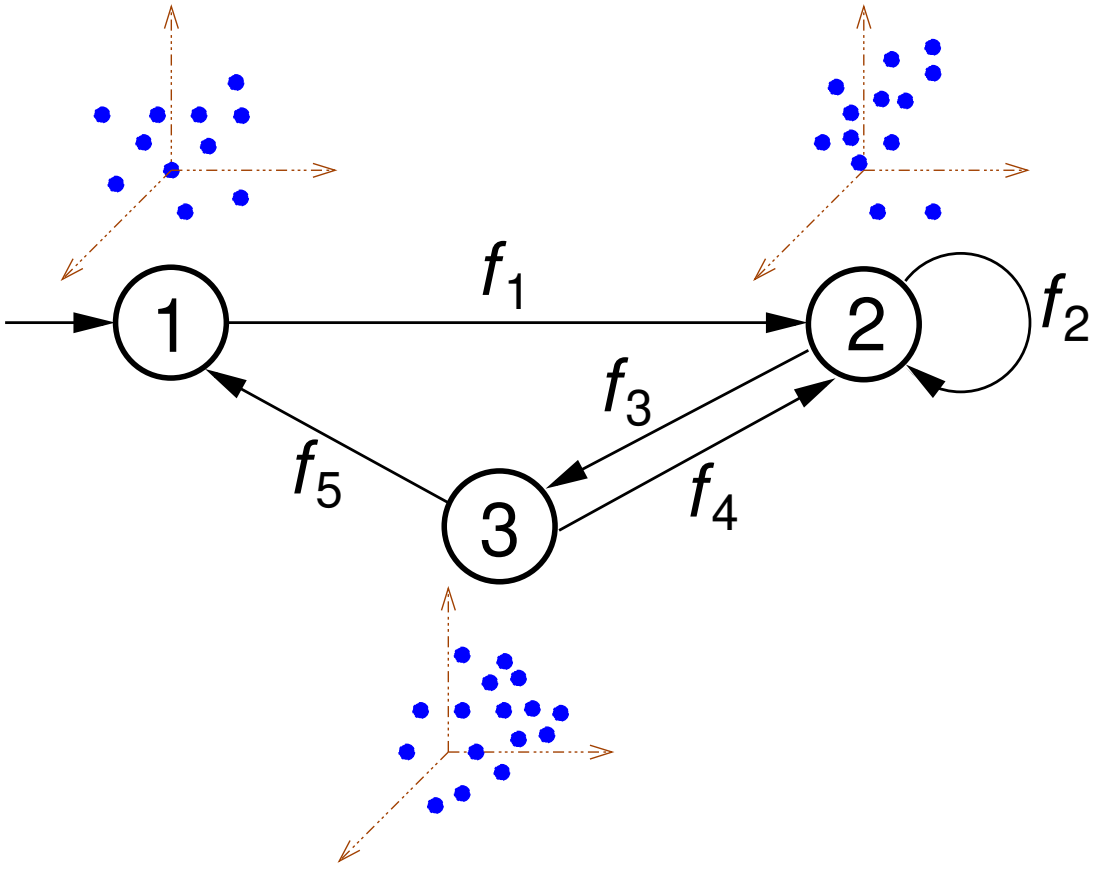
Geometric Picture

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



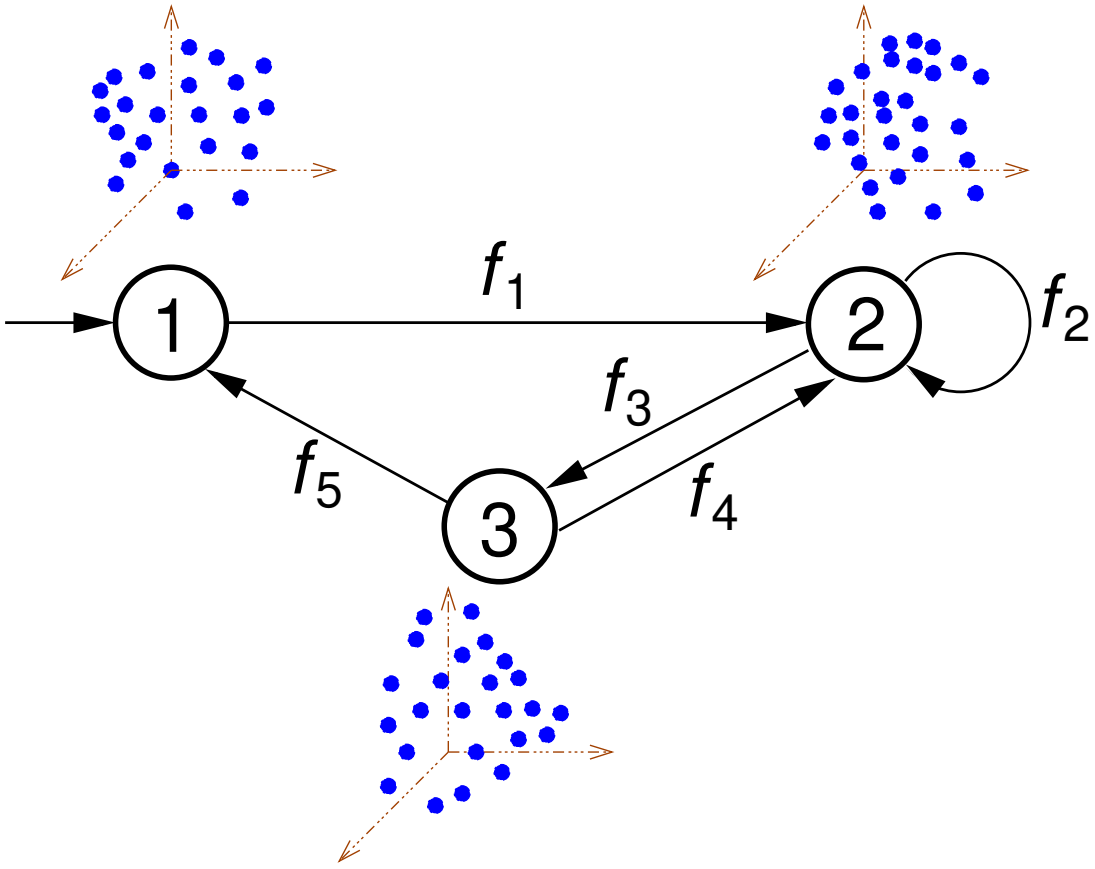
Geometric Picture

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



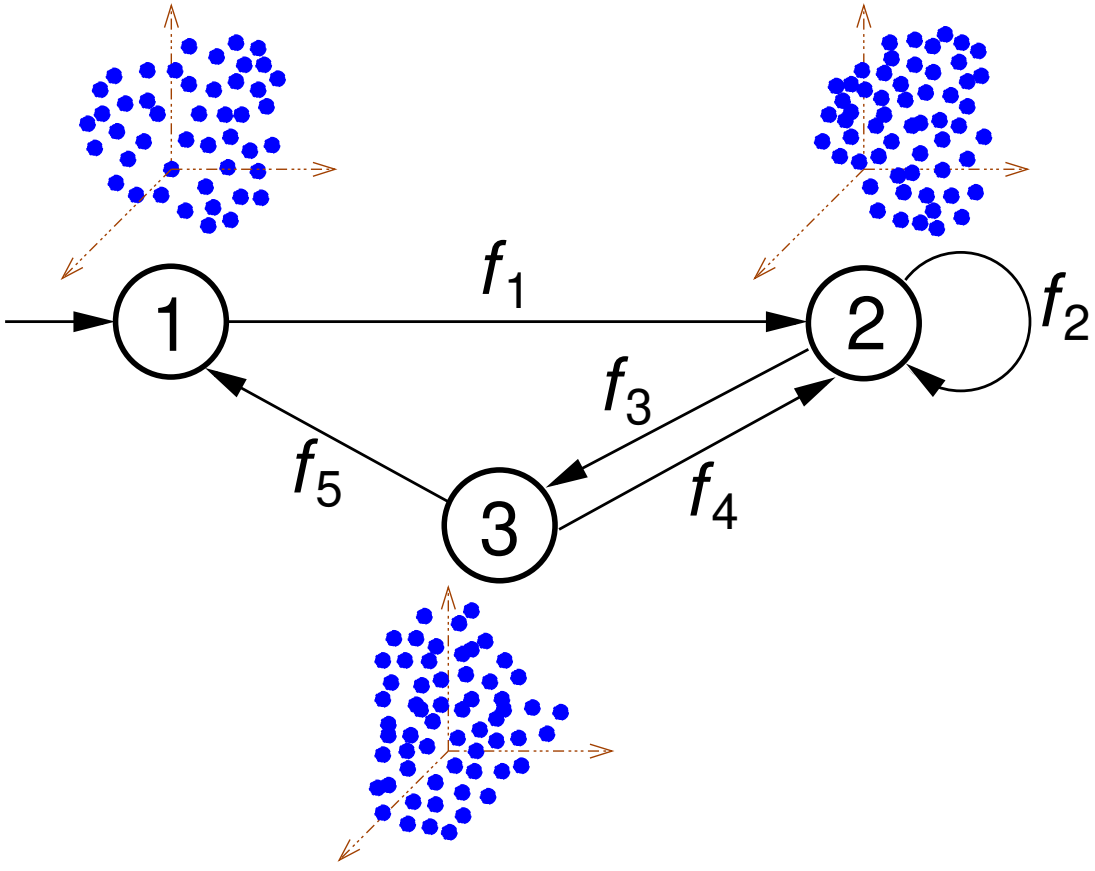
Geometric Picture

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



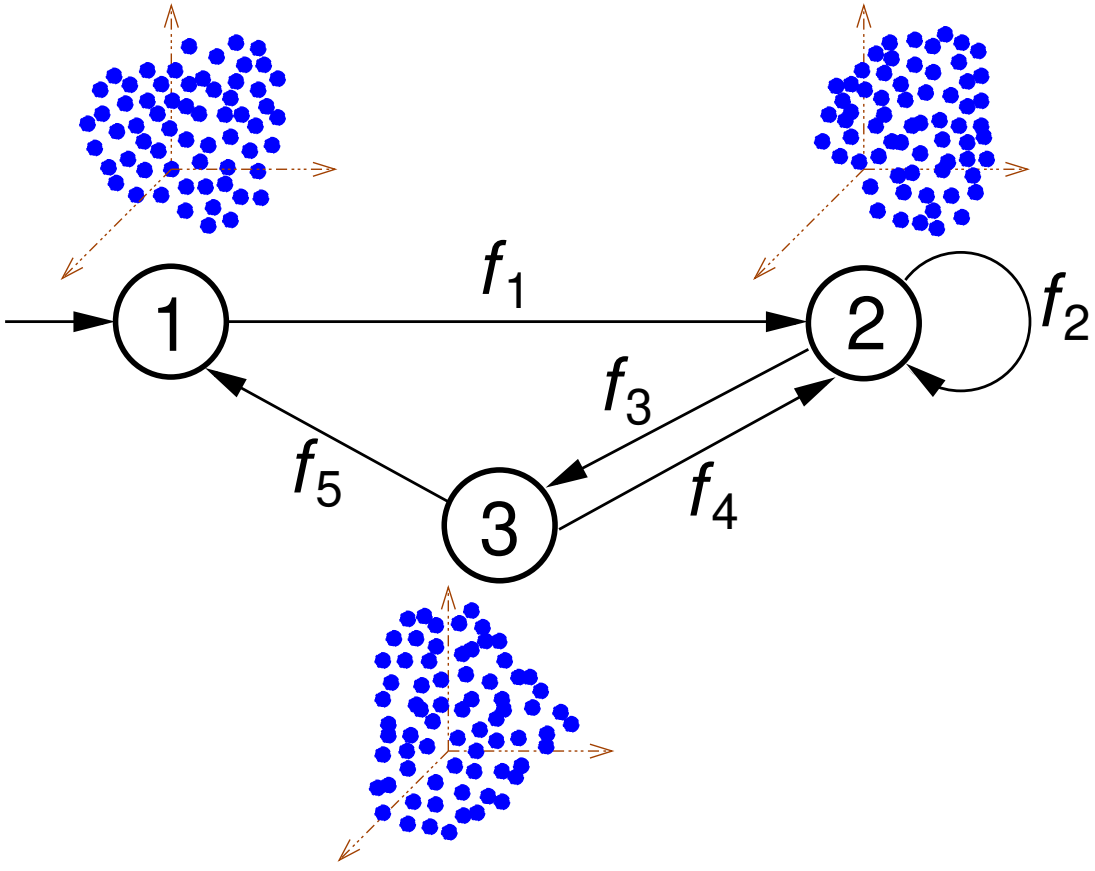
Geometric Picture

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



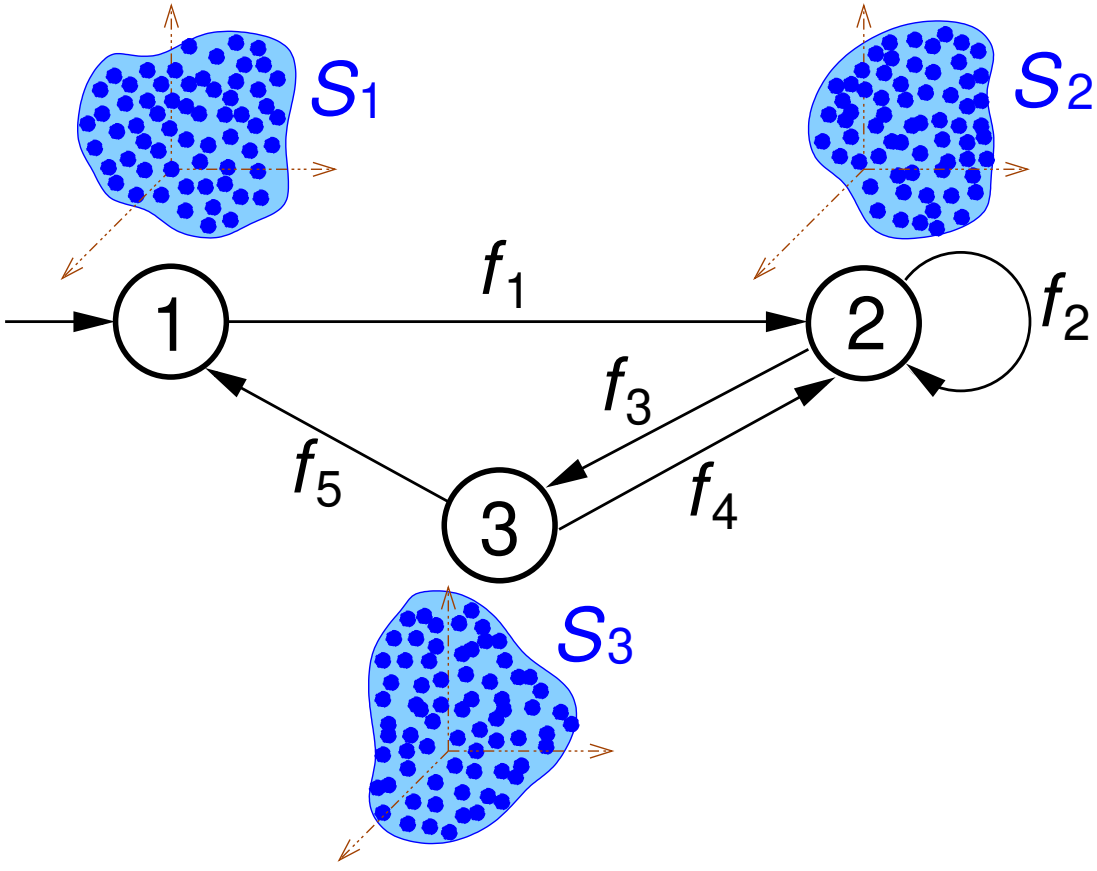
Geometric Picture

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



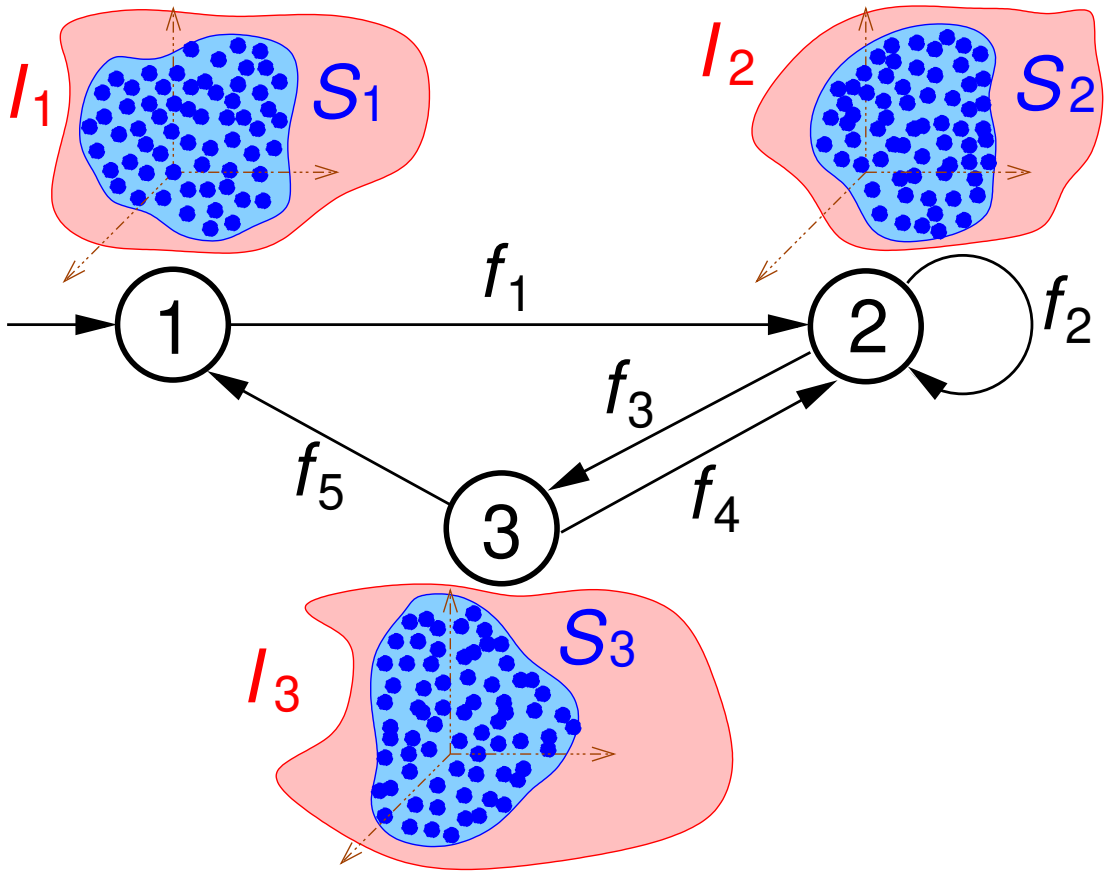
Geometric Picture

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



Geometric Picture

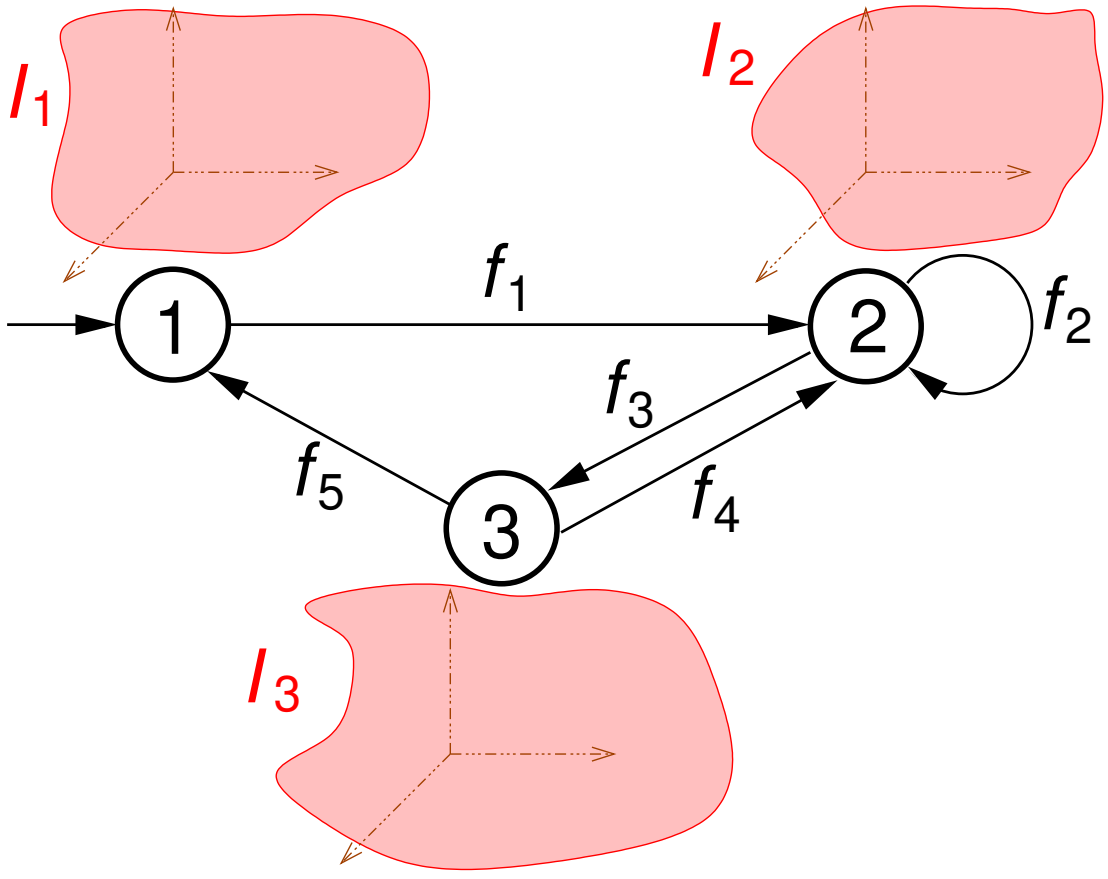
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **invariant**

Geometric Picture

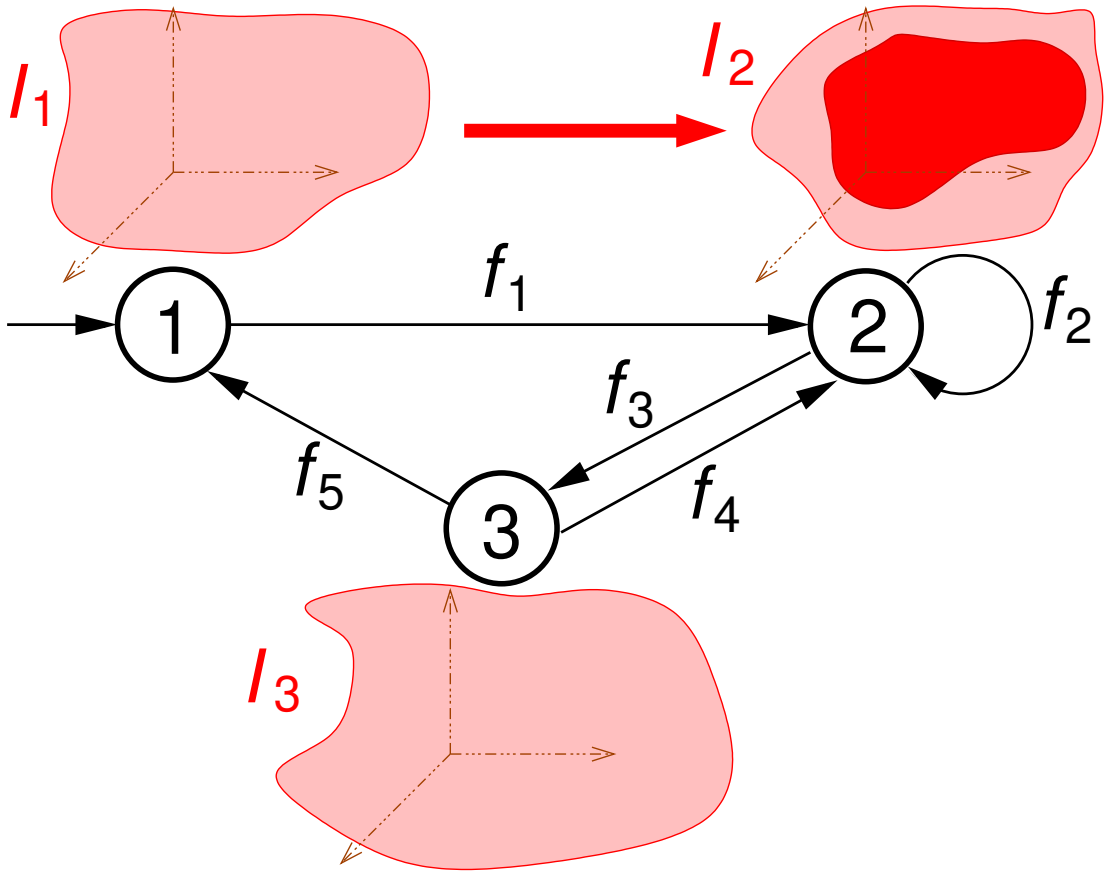
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle I_1, I_2, I_3 \rangle$ is an **invariant**

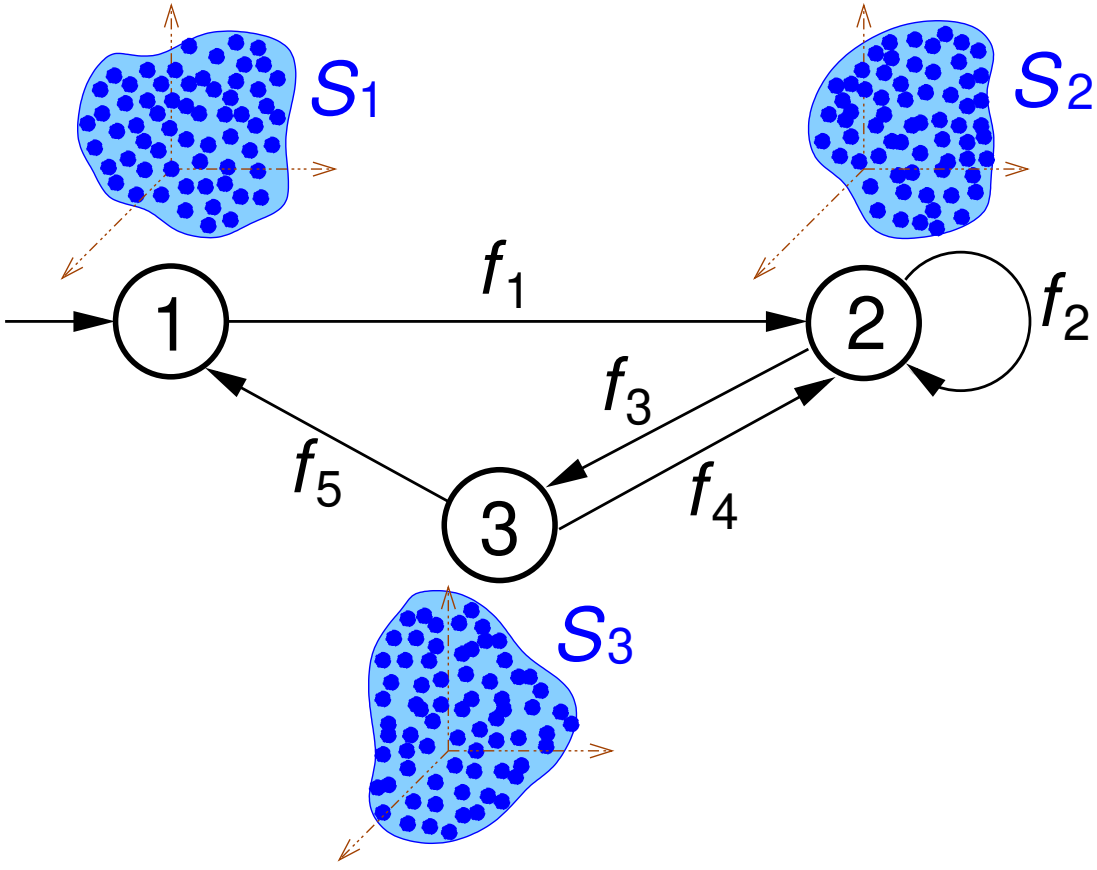
Geometric Picture

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

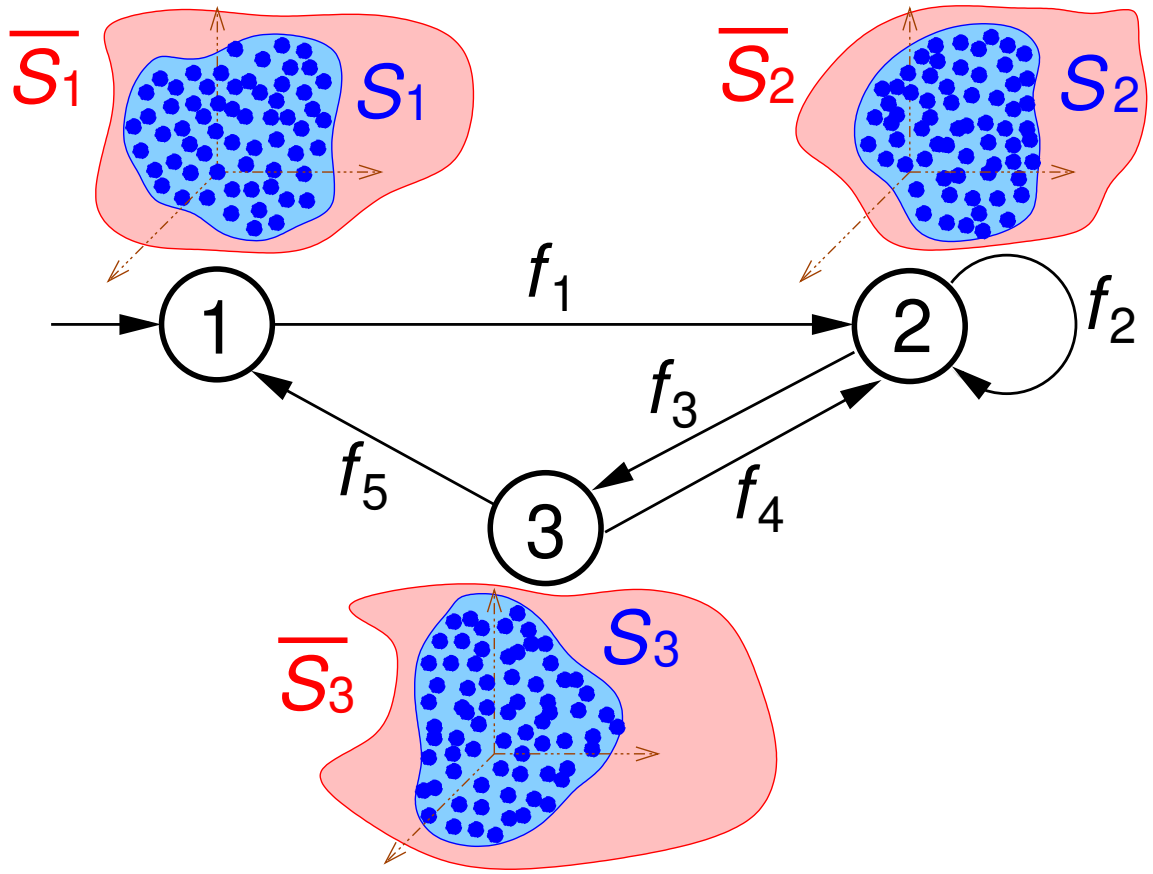


$\langle I_1, I_2, I_3 \rangle$ is an **inductive invariant**

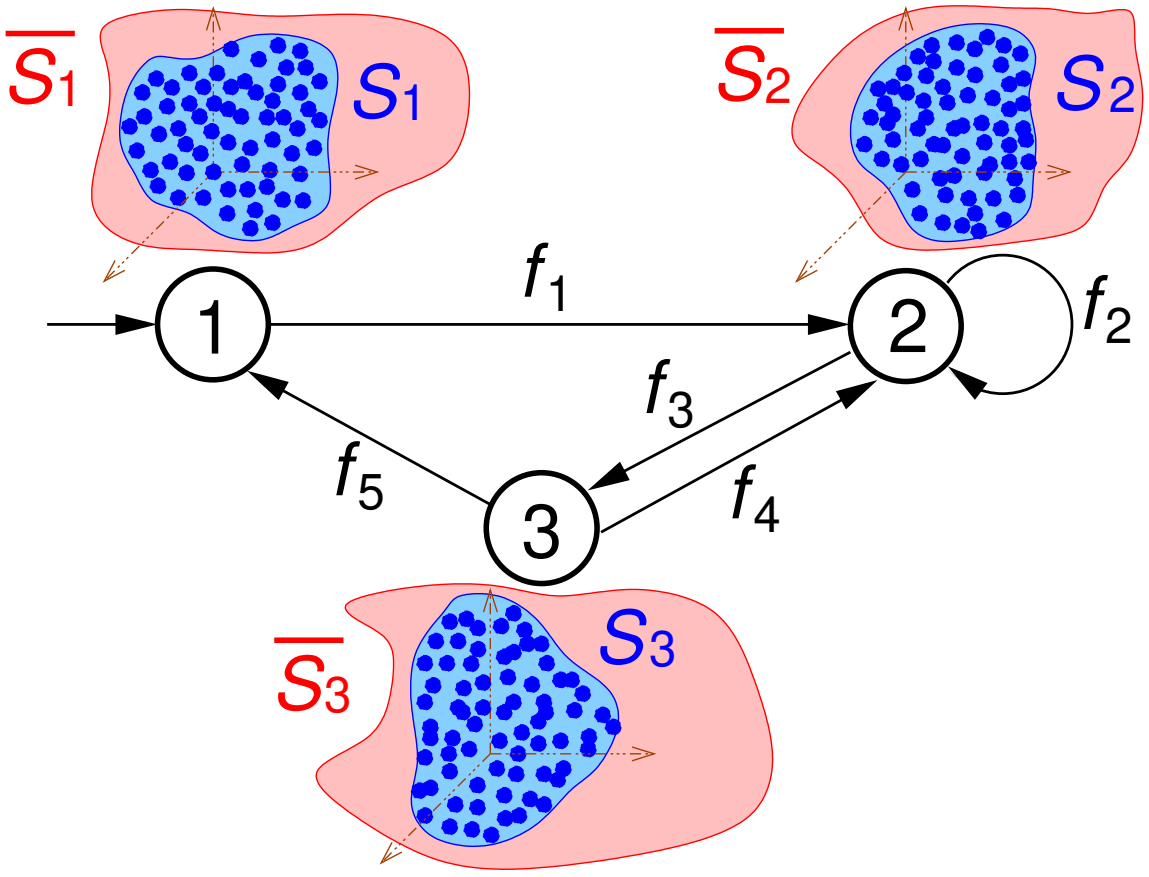
Computing Strongest Polynomial Invariants



Computing Strongest Polynomial Invariants

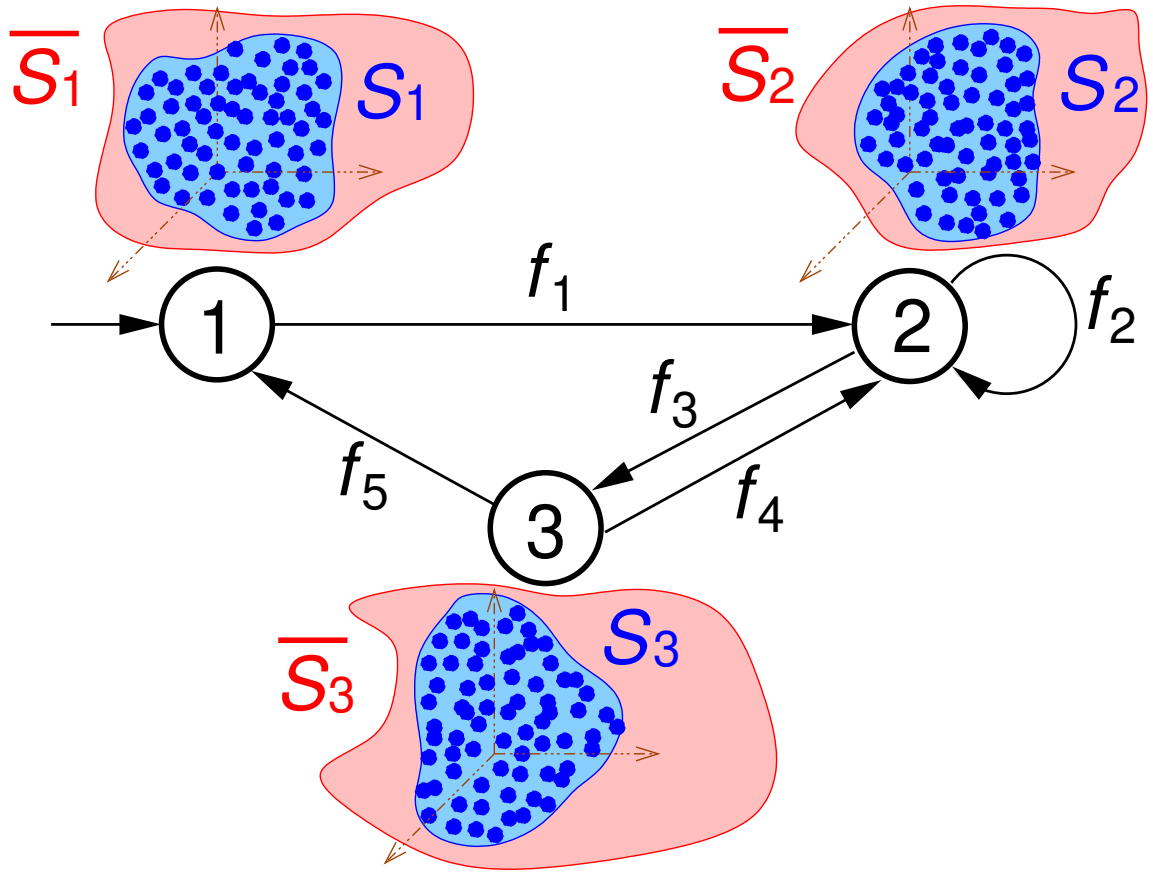


Computing Strongest Polynomial Invariants



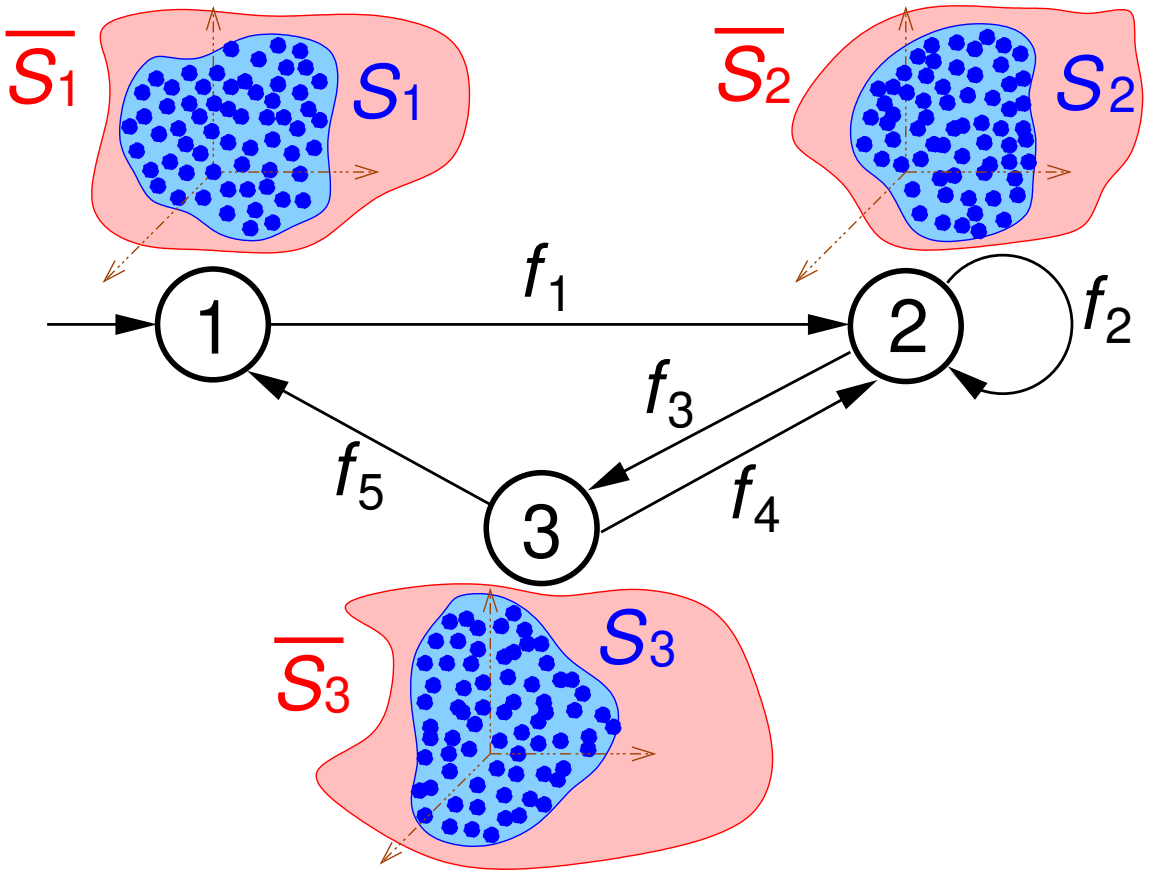
$f_1(S_1) \subseteq S_2$

Computing Strongest Polynomial Invariants



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

Computing Strongest Polynomial Invariants



$$f_1(S_1) \subseteq S_2 \implies f_1(\overline{S_1}) \subseteq \overline{S_2}$$

Invariants are inductive because affine functions are continuous

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 . . .

Affine Invariants for Affine Programs

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 that computes, for any given affine program over $\mathbb{Q}$*, its* **strongest affine inductive invariant**.

Polynomial Invariants for Affine Programs

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 $\mathcal{O}(nk^{3d})$.

Theorem (ICALP 2004)

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

Finding all polynomial invariants

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

Finding all polynomial invariants

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 problem whether or not the set of all valid polynomial relations can be computed, not just the ones of some given form"

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.

- Algorithm computes for each location the set of **all polynomial relations** among program variables that hold 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.

- Algorithm computes for each location the set of **all polynomial relations** among program variables that hold whenever control reaches that location
- We represent this set of relations using a **finite basis** 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.

- Algorithm computes for each location the set of **all polynomial relations** among program variables that hold whenever control reaches that location
- We represent this set of relations using a **finite basis** of polynomial equalities
- Dually, the algorithm computes for each location the **smallest algebraic set** containing the set of reachable states:

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.

- Algorithm computes for each location the set of **all polynomial relations** among program variables that hold whenever control reaches that location
- We represent this set of relations using a **finite basis** of polynomial equalities
- Dually, the algorithm computes for each location the **smallest algebraic set** containing the set of reachable states:
  - **algebraic sets** are defined by conjunctions of polynomial equalities

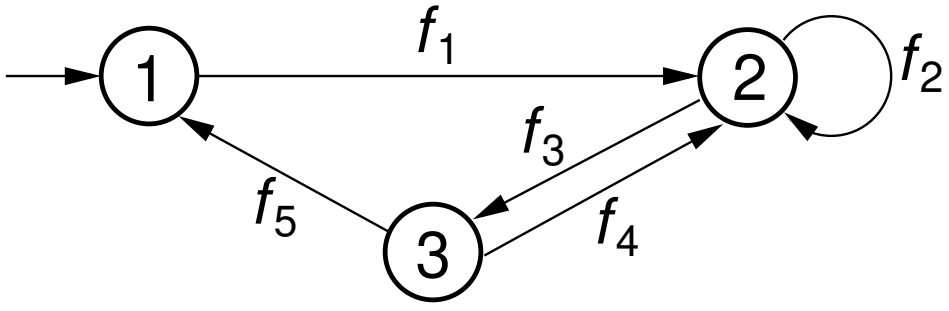
Strongest Polynomial Invariants @ LICS 2018

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

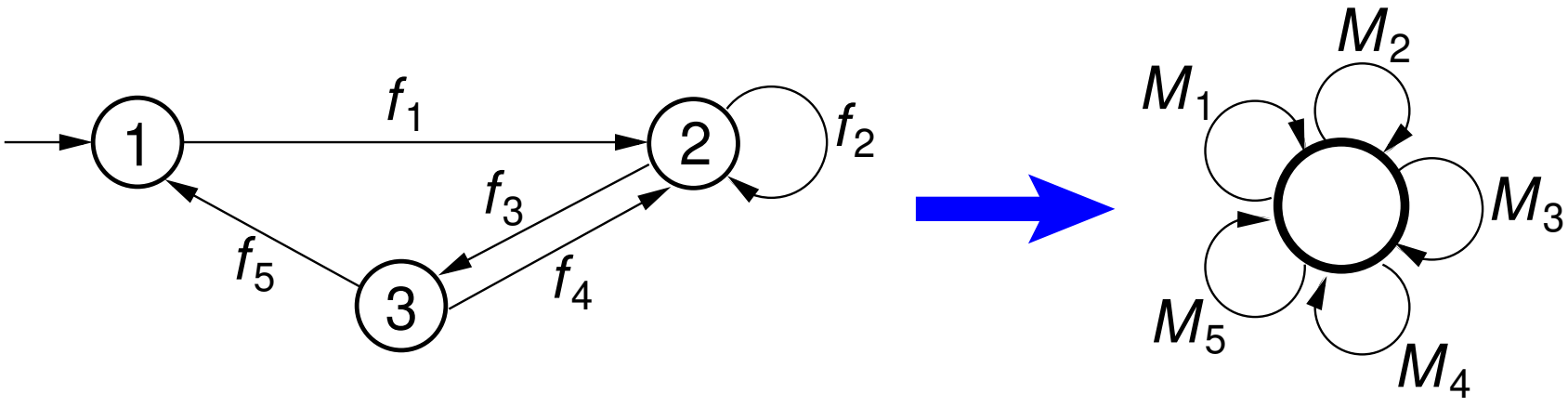
There is an algorithm which computes, for any given affine program over $\mathbb{Q}$, its strongest polynomial inductive invariant.

- Algorithm computes for each location the set of **all polynomial relations** among program variables that hold whenever control reaches that location
- We represent this set of relations using a **finite basis** of polynomial equalities
- Dually, the algorithm computes for each location the **smallest algebraic set** containing the set of reachable states:
  - **algebraic sets** are defined by conjunctions of polynomial equalities
  - Smallest algebraic set = **Zariski closure**

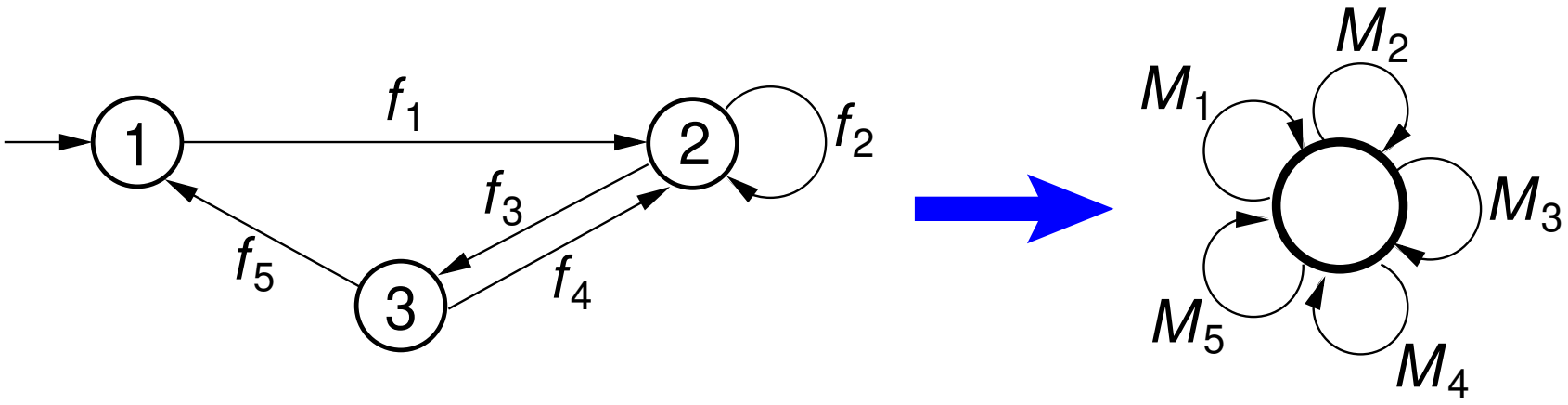
From Affine Programs to Linear Semigroups



From Affine Programs to Linear Semigroups



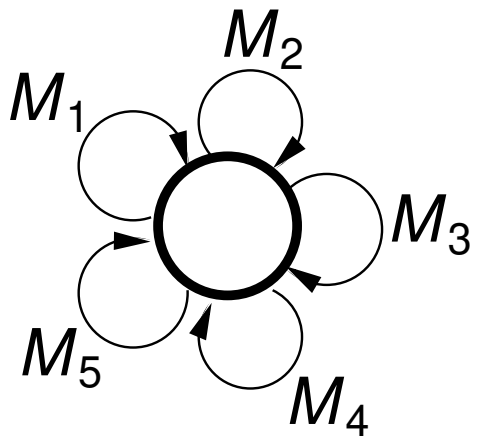
From Affine Programs to Linear Semigroups



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

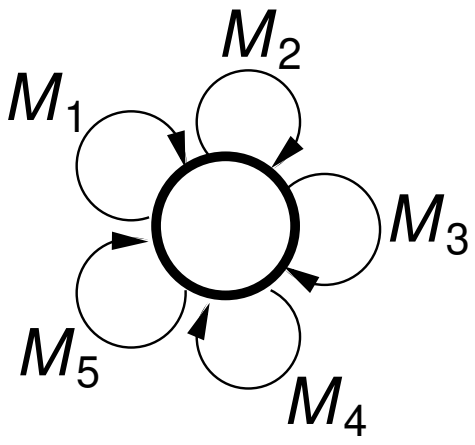
Zariski Closure of Linear Semigroups

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



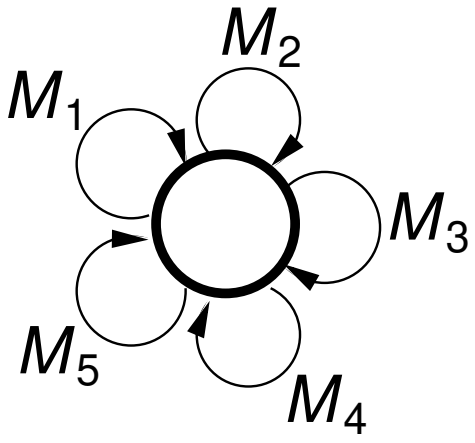
Zariski Closure of Linear Semigroups

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



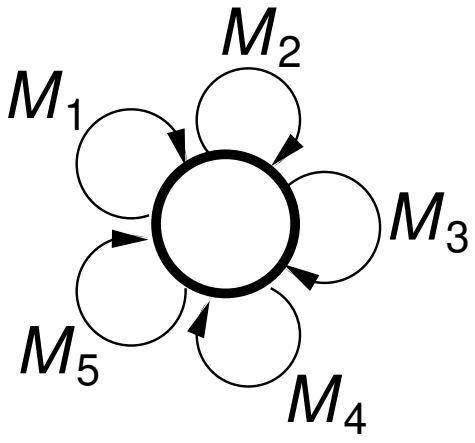
Zariski Closure of Linear Semigroups

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

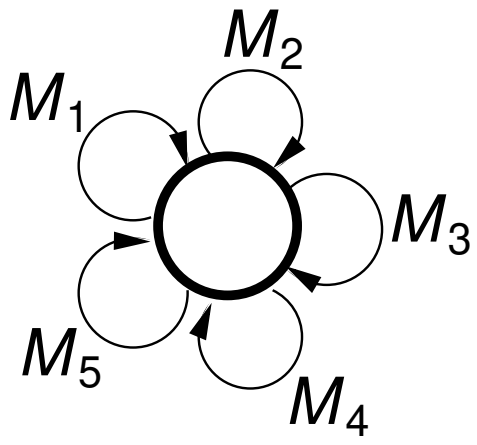


Theorem (Hrushovski, Ouaknine, Pouly, W. 18)

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

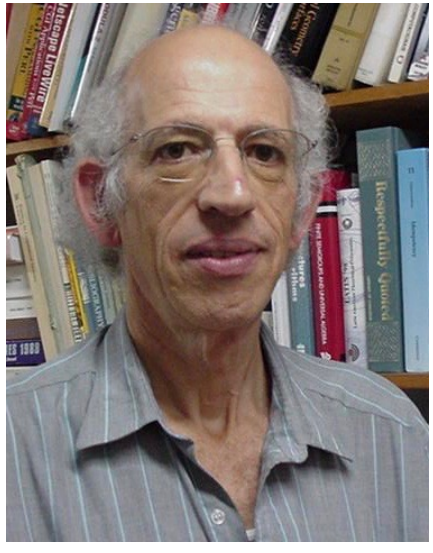
ON FINITE SEMIGROUPS OF MATRICES*

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 6×6 *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 6×6 *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 6×6 *integer matrices* $M_1, \ldots, M_k$ *such that the membership problem "*$M \in \langle M_1, \ldots, M_k \rangle$*?" is* **undecidable**.

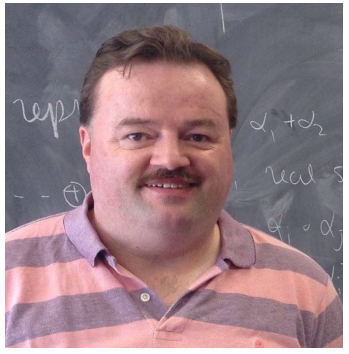
Mortality: Is the zero matrix contained in the semigroup generated by a given set of $n \times n$ matrices with integer entries?

Theorem (Paterson 1970)

The mortality problem is **undecidable** *for* 3×3 *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.
© 2005 Elsevier Ltd. All rights reserved.

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

Some Ingredients

Theorem (Schur 1911)

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

Some Ingredients

Theorem (Schur 1911)

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

Theorem (Masser 1988)

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

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

Multiplicative Relations

Write $A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -4 \end{pmatrix}$:

Multiplicative Relations

Write $A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -4 \end{pmatrix}$:

$$\overline{\{A^n : n \in \mathbb{Z}\}} \quad = \quad \overline{\{A^n : n \in \mathbb{N}\}}$$

Multiplicative Relations

Write $A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -4 \end{pmatrix}$:

$$\begin{aligned} \overline{\{A^n : n \in \mathbb{Z}\}} &= \overline{\{A^n : n \in \mathbb{N}\}} \\ &= \overline{\left\{ \begin{pmatrix} 2^n & n2^{n-1} & 0 \\ 0 & 2^n & 0 \\ 0 & 0 & (-4)^n \end{pmatrix} : n \in \mathbb{N} \right\}} \end{aligned}$$

Multiplicative Relations

Write $A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -4 \end{pmatrix}$:

$$\begin{aligned} \overline{\{A^n : n \in \mathbb{Z}\}} &= \overline{\{A^n : n \in \mathbb{N}\}} \\ &= \overline{\left\{ \begin{pmatrix} 2^n & n2^{n-1} & 0 \\ 0 & 2^n & 0 \\ 0 & 0 & (-4)^n \end{pmatrix} : n \in \mathbb{N} \right\}} \\ &= \left\{ \begin{pmatrix} x & y & 0 \\ 0 & x & 0 \\ 0 & 0 & x^2 \end{pmatrix}, \begin{pmatrix} x & y & 0 \\ 0 & x & 0 \\ 0 & 0 & -x^2 \end{pmatrix} : x, y \in \mathbb{R}, x \neq 0 \right\} \end{aligned}$$

Multiplicative Relations

Write $A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -4 \end{pmatrix}$:

$$\begin{aligned} \overline{\{A^n : n \in \mathbb{Z}\}} &= \overline{\{A^n : n \in \mathbb{N}\}} \\ &= \overline{\left\{ \begin{pmatrix} 2^n & n2^{n-1} & 0 \\ 0 & 2^n & 0 \\ 0 & 0 & (-4)^n \end{pmatrix} : n \in \mathbb{N} \right\}} \\ &= \left\{ \begin{pmatrix} x & y & 0 \\ 0 & x & 0 \\ 0 & 0 & x^2 \end{pmatrix}, \begin{pmatrix} x & y & 0 \\ 0 & x & 0 \\ 0 & 0 & -x^2 \end{pmatrix} : x, y \in \mathbb{R}, x \neq 0 \right\} \end{aligned}$$

Zariski closure is determined by **multiplicative relationships** among eigenvalues.

Categorical Structure in Linear Semigroups

Given algebraic semigroup $\boldsymbol{S} \subseteq M_n(\mathbb{Q})$ and $r \in \mathbb{N}$, define

$$\boldsymbol{S}_r := \{A \in \boldsymbol{S} : \operatorname{rank}(A) = r\}.$$

Categorical Structure in Linear Semigroups

Given algebraic semigroup $\boldsymbol{S} \subseteq M_n(\mathbb{Q})$ and $r \in \mathbb{N}$, define

$$\boldsymbol{S}_r := \{A \in \boldsymbol{S} : \operatorname{rank}(A) = r\}.$$

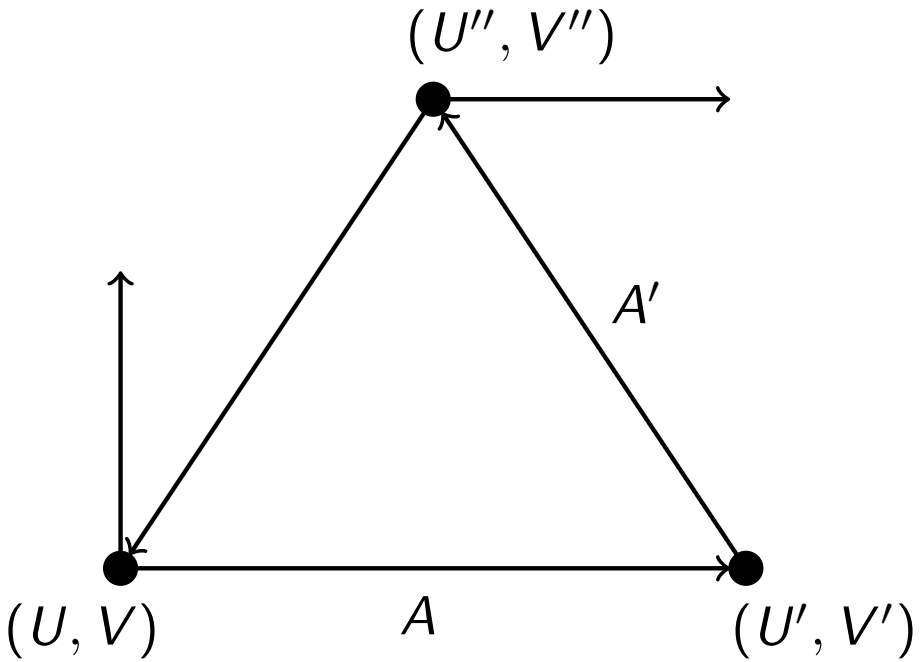
Not a semigroup in general, but consider $\boldsymbol{S}_r$ as a **category**:

Categorical Structure in Linear Semigroups

Given algebraic semigroup $\boldsymbol{S} \subseteq M_n(\mathbb{Q})$ and $r \in \mathbb{N}$, define

$$\boldsymbol{S}_r := \{A \in \boldsymbol{S} : \operatorname{rank}(A) = r\}.$$

Not a semigroup in general, but consider $\boldsymbol{S}_r$ as a **category**:

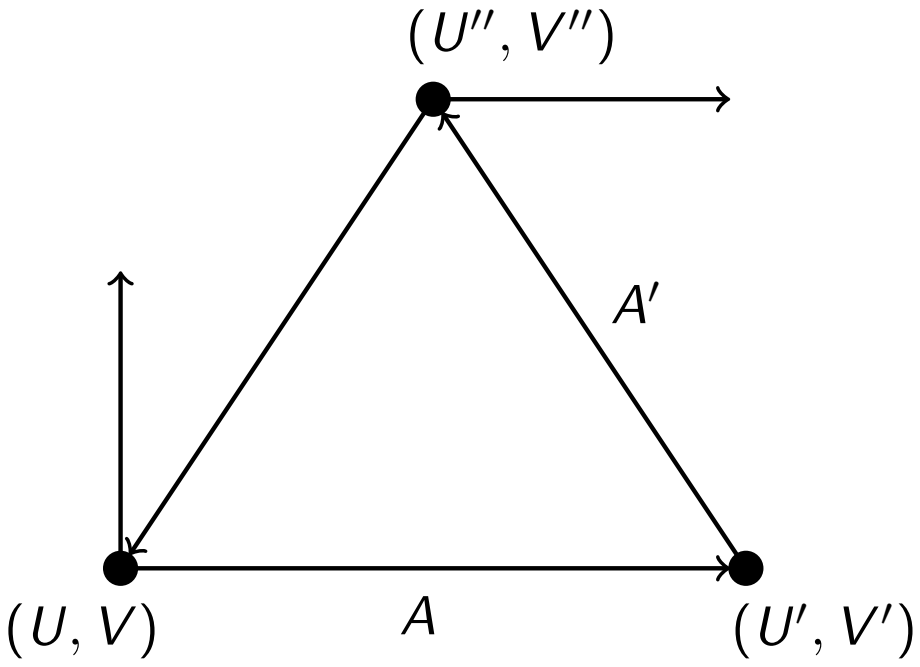


Categorical Structure in Linear Semigroups

Given algebraic semigroup $\boldsymbol{S} \subseteq M_n(\mathbb{Q})$ and $r \in \mathbb{N}$, define

$$\boldsymbol{S}_r := \{A \in \boldsymbol{S} : \operatorname{rank}(A) = r\}.$$

Not a semigroup in general, but consider $\boldsymbol{S}_r$ as a **category**:



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

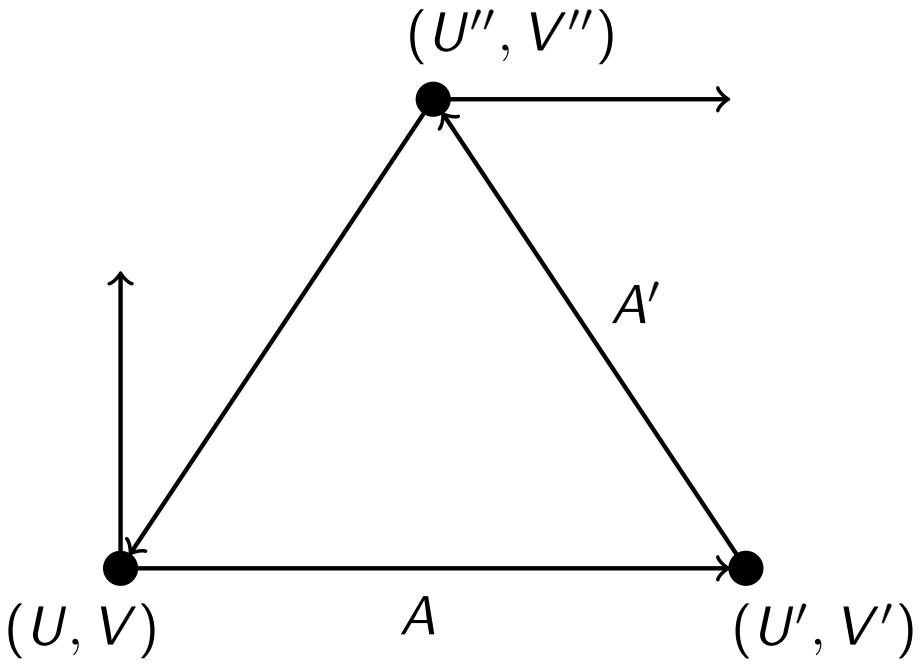
- $U \cap V = 0$
- $\dim(U) = n - r$, $\dim(V) = r$

Categorical Structure in Linear Semigroups

Given algebraic semigroup $\boldsymbol{S} \subseteq M_n(\mathbb{Q})$ and $r \in \mathbb{N}$, define

$$\boldsymbol{S}_r := \{A \in \boldsymbol{S} : \operatorname{rank}(A) = r\}.$$

Not a semigroup in general, but consider $\boldsymbol{S}_r$ as a **category**:



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

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

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

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

Properties of $\mathcal{C}_r$

- Each non-trivial SCC is a groupoid.

Properties of $\mathcal{C}_r$

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

Properties of $\mathcal{C}_r$

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

Roughly Speaking . . .

We generalise the algorithm of Derksen, Jeandel, and Koiran from finitely generated groups to groupoids with algebraic sets of generators.

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

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.

From Affine Programs to Polynomial Programs

Equivalence of Deterministic Top-Down Tree-to-String Transducers Is Decidable

HELMUT SEIDL, Technical University of Munich
SEBASTIAN MANETH, Universität of Bremen
GREGOR KEMPER, Technical University of Munich

> *" [. . .] we introduce polynomial transducers and prove that for these, equivalence can be certified by means of an inductive polynomial invariant. This allows us to construct two semi-algorithms, one searching for an invariant and the other for a witness of non-equivalence [. . .]"*

Undecidability for Polynomial Programs

Undecidability for Polynomial Programs

Theorem (Dufourd, Finkel, Schnoebelen 1998)

The boundedness problem for reset vector addition systems is undecidable.

Undecidability for Polynomial Programs

Theorem (Dufourd, Finkel, Schnoebelen 1998)

The boundedness problem for reset vector addition systems is undecidable.

Theorem (Hrushovski, Ouaknine, Pouly, W. 23)

There is no algorithm that computes the Zariski closure of the reachable set of a polynomial program.

Undecidability for Polynomial Programs

Theorem (Dufourd, Finkel, Schnoebelen 1998)

The boundedness problem for reset vector addition systems is undecidable.

Theorem (Hrushovski, Ouaknine, Pouly, W. 23)

There is no algorithm that computes the Zariski closure of the reachable set of a polynomial program.

- Simulate reset VAS by polynomial program:

Undecidability for Polynomial Programs

Theorem (Dufourd, Finkel, Schnoebelen 1998)

The boundedness problem for reset vector addition systems is undecidable.

Theorem (Hrushovski, Ouaknine, Pouly, W. 23)

There is no algorithm that computes the Zariski closure of the reachable set of a polynomial program.

- Simulate reset VAS by polynomial program:
- Represent VAS configuration $(a_1, \ldots, a_d)$ "projectively" by configuration $(z, a_1 z, \ldots, a_d z)$, $z \neq 0$: of polynmial program

$$\mathrm{dec}(1): \quad (z, a_1, \ldots, a_d) := (z a_1, (a_1 - z) a_1, a_2 a_1, \ldots, a_d a_1)$$

Undecidability for Polynomial Programs

Theorem (Dufourd, Finkel, Schnoebelen 1998)

The boundedness problem for reset vector addition systems is undecidable.

Theorem (Hrushovski, Ouaknine, Pouly, W. 23)

There is no algorithm that computes the Zariski closure of the reachable set of a polynomial program.

- Simulate reset VAS by polynomial program:
- Represent VAS configuration $(a_1, \ldots, a_d)$ "projectively" by configuration $(z, a_1 z, \ldots, a_d z)$, $z \neq 0$: of polynmial program

$$\mathrm{dec}(1): \quad (z, a_1, \ldots, a_d) := (z a_1, (a_1 - z) a_1, a_2 a_1, \ldots, a_d a_1)$$

- VAS is bounded iff Zariski closure has dimension at most one

Postscript: A Challenge in Program Analysis

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

[. . .] use **inequality relationships** *to determine at compile time whether the value of an expression is within a specified range. This includes compile-time overflow, integer subrange, and array bound checking.*

Postscript: A Challenge in Program Analysis

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

[. . .] use **inequality relationships** *to determine at compile time whether the value of an expression is within a specified range. This includes compile-time overflow, integer subrange, and array bound checking.*

Compute inductive invariants determined by linear and polynomial inequalities?

The Monniaux Problem

P. Cousot

N. Halbwachs

D. Monniaux

"Forty years of research on convex polyhedral invariants have focused, on the one hand, on identifying "easier" subclasses, on the other hand on heuristics for finding general convex polyhedra. These heuristics are however not guaranteed to find polyhedral inductive invariants when they exist. To our best knowledge, the existence of polyhedral inductive invariants has never been proved to be undecidable."

– David Monniaux, Acta Inf. 2019