On the Monniaux Problem in Abstract Interpretation

Nathanaël Fijalkow, Engel Lefaucheux, Pierre Ohlmann,
Joël Ouaknine, Amaury Pouly and James Worrell

LaBRI, Max Planck Institute for Software Systems, IRIF, Oxford University

IRISA, November 2019

The concept of an invariant is one of the most important in mathematics.

Encyclopedia of Mathematics, Kluwer, 2002

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**
2. **M**X $\rightarrow$ **M**XX

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

   You can replace any **III** with a **U**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

   You can replace any **III** with a **U**
   Example: **MUIIIU** becomes **MUUU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

   You can replace any **III** with a **U**
   Example: **MUIIIU** becomes **MUUU**

4. X**UU**Y $\rightarrow$ XY

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

   You can replace any **III** with a **U**
   Example: **MUIIIU** becomes **MUUU**

4. X**UU**Y $\rightarrow$ XY

   You can remove any **UU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

   You can replace any **III** with a **U**
   Example: **MUIIIU** becomes **MUUU**

4. X**UU**Y $\rightarrow$ XY

   You can remove any **UU**
   Example: **MUUU** becomes **MU**

The MU Puzzle

A string-rewriting system using letters **M**, **I**, and **U**

0. We start with **MI**
1. X**I** $\rightarrow$ X**IU**

   You can add **U** at the end of any string ending in **I**
   Example: **MI** becomes **MIU**

2. **M**X $\rightarrow$ **M**XX

   You can double the string after the **M**
   Example: **MIU** becomes **MIUIU**

3. X**III**Y $\rightarrow$ X**U**Y

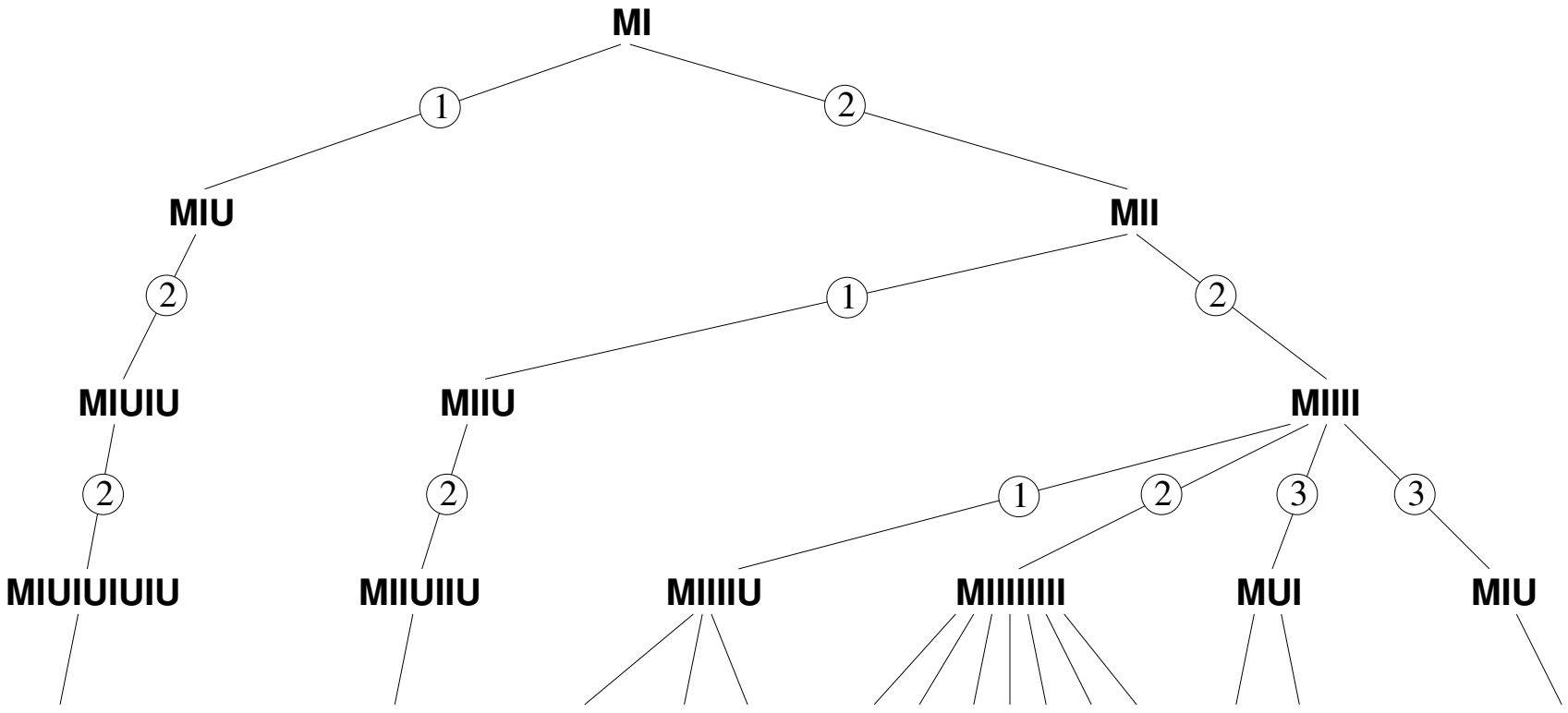
   You can replace any **III** with a **U**
   Example: **MUIIIU** becomes **MUUU**

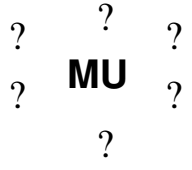
4. X**UU**Y $\rightarrow$ XY

   You can remove any **UU**
   Example: **MUUU** becomes **MU**

Starting from **MI**, the goal is to produce **MU**

Can It Be Solved?



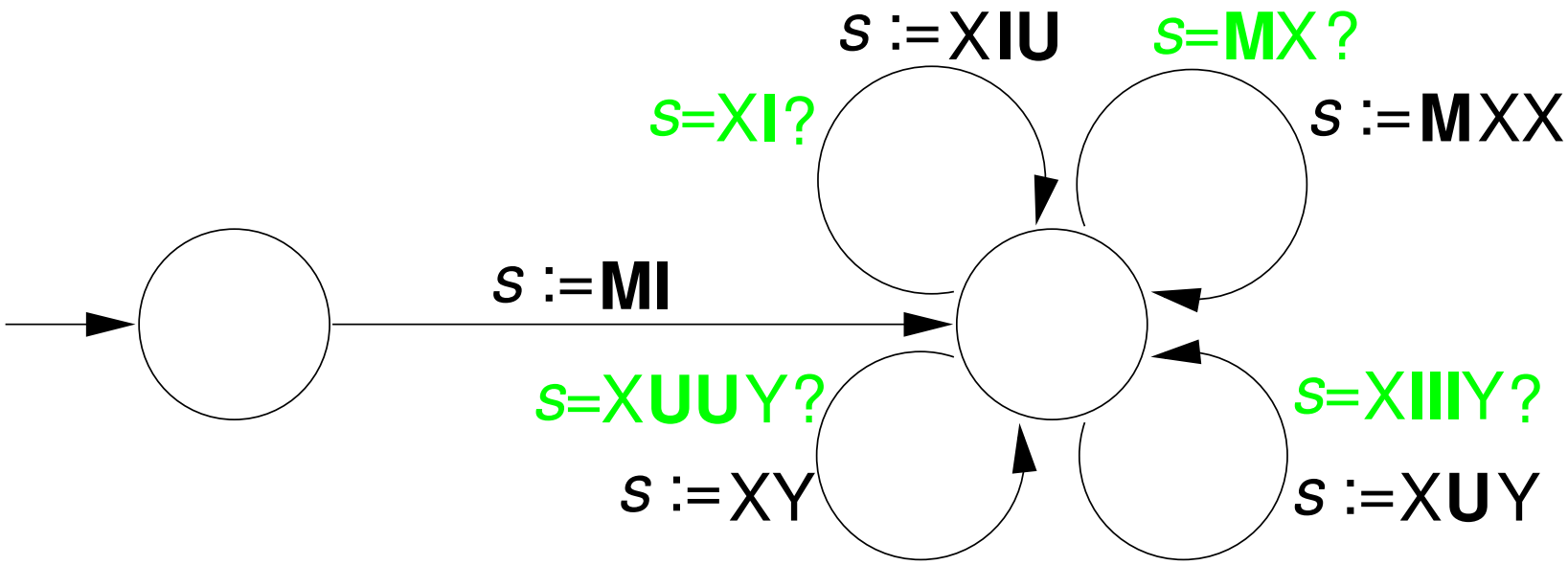


Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
    choose
        $\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
        $\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
        $\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
        $\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

Can This (Nondeterministic) Program Halt?

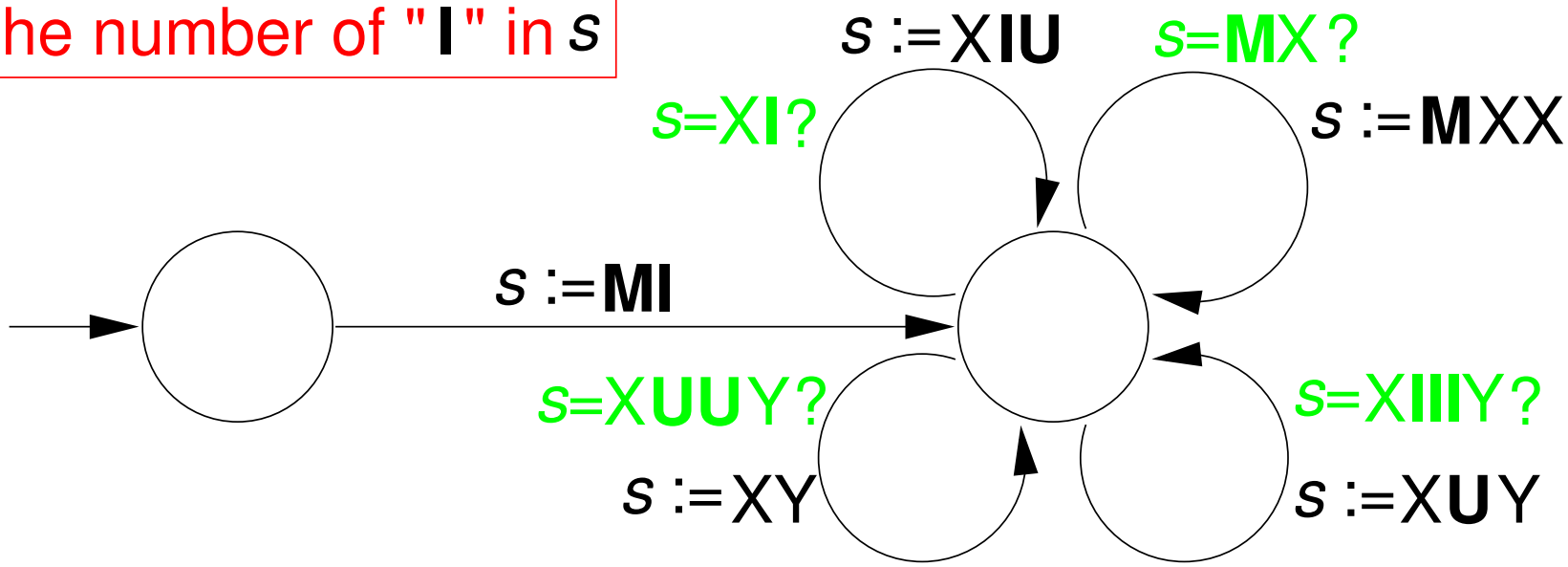
$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

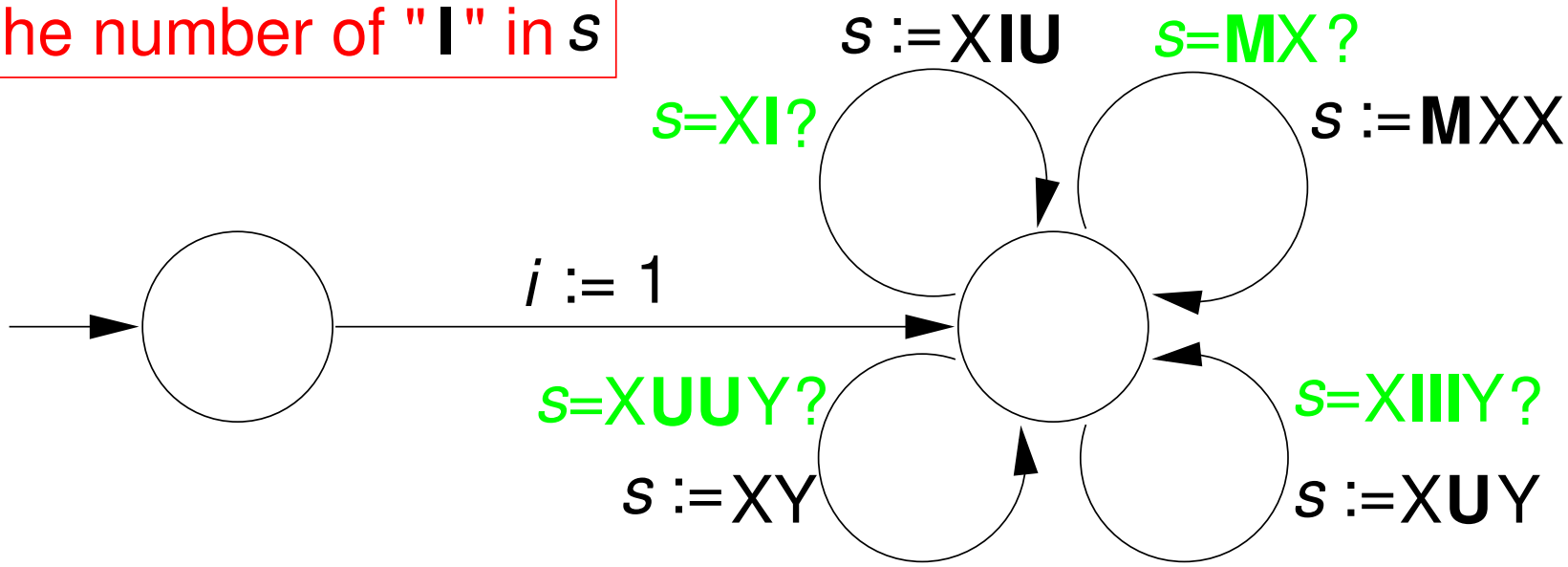
introduce "i" to count the number of "**I**" in s



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
  choose
    $\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
    $\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
    $\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
    $\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

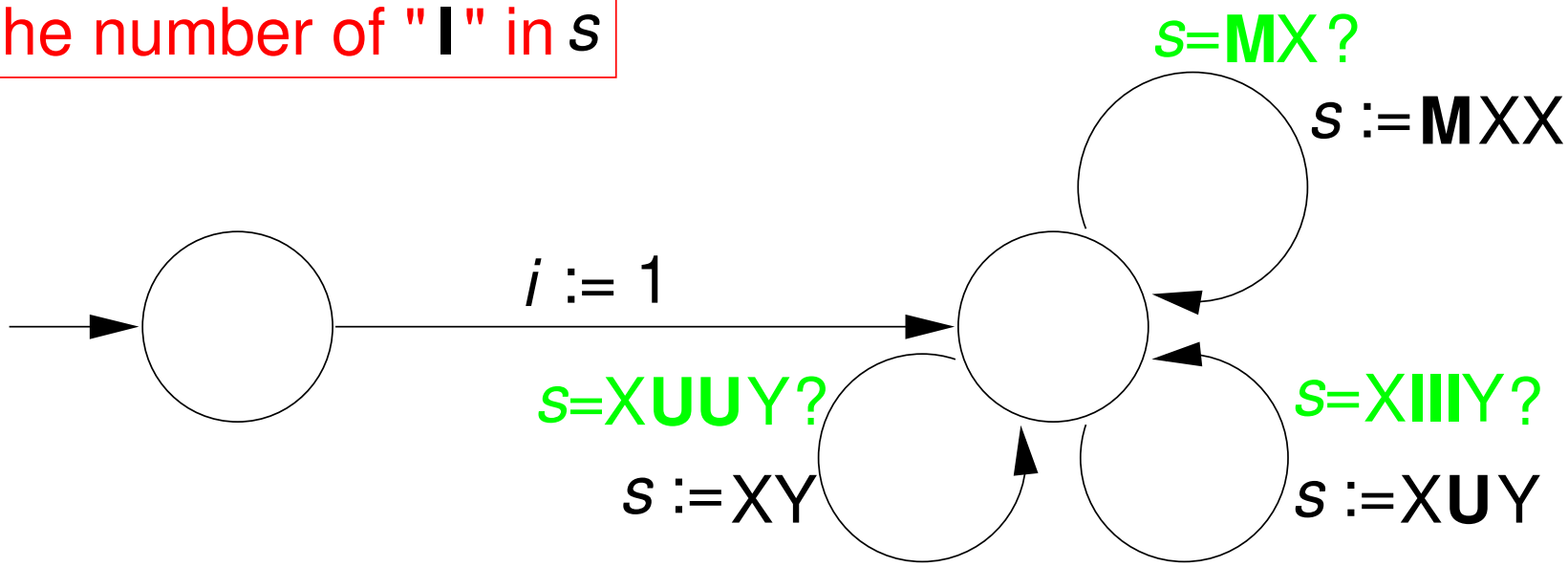
introduce "i" to count the number of "**I**" in s



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

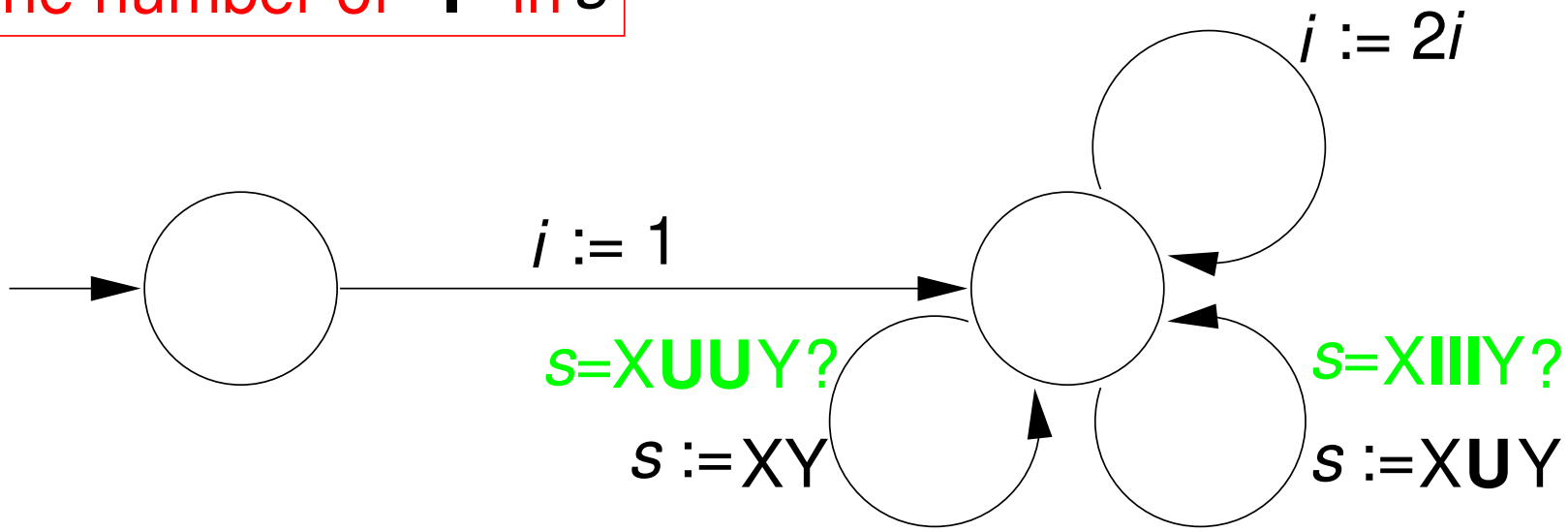
introduce "i" to count the number of "**I**" in s



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

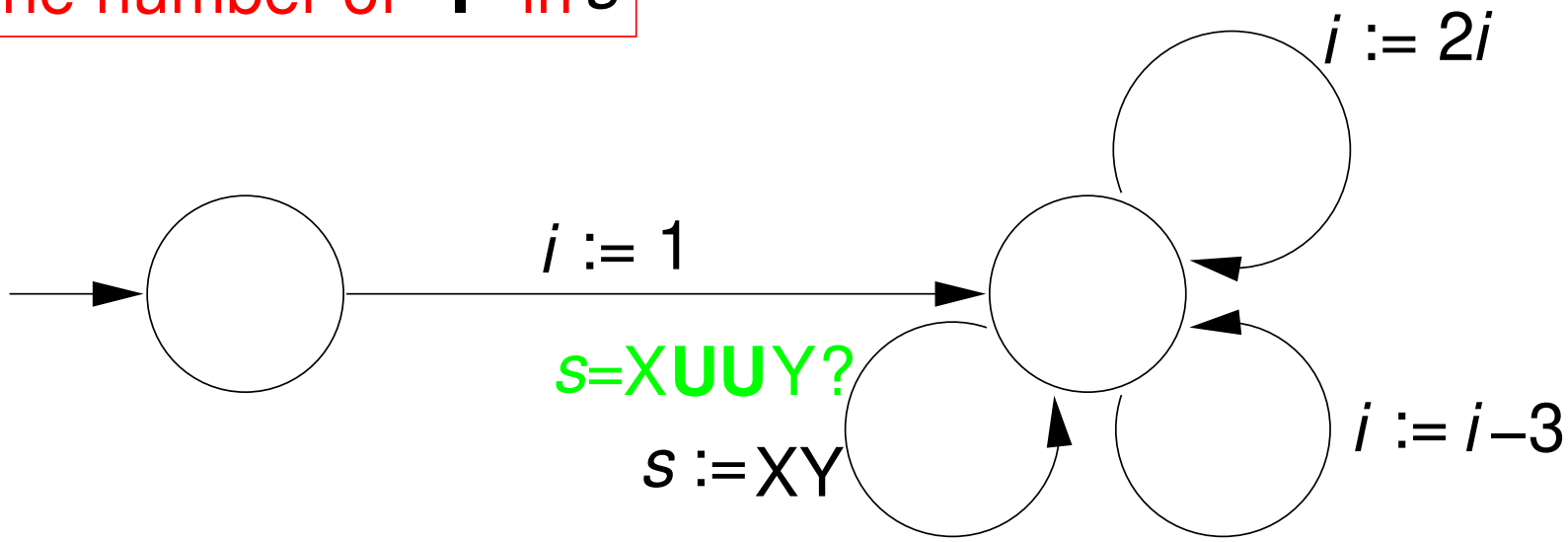
introduce "i" to count the number of "**I**" in s



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

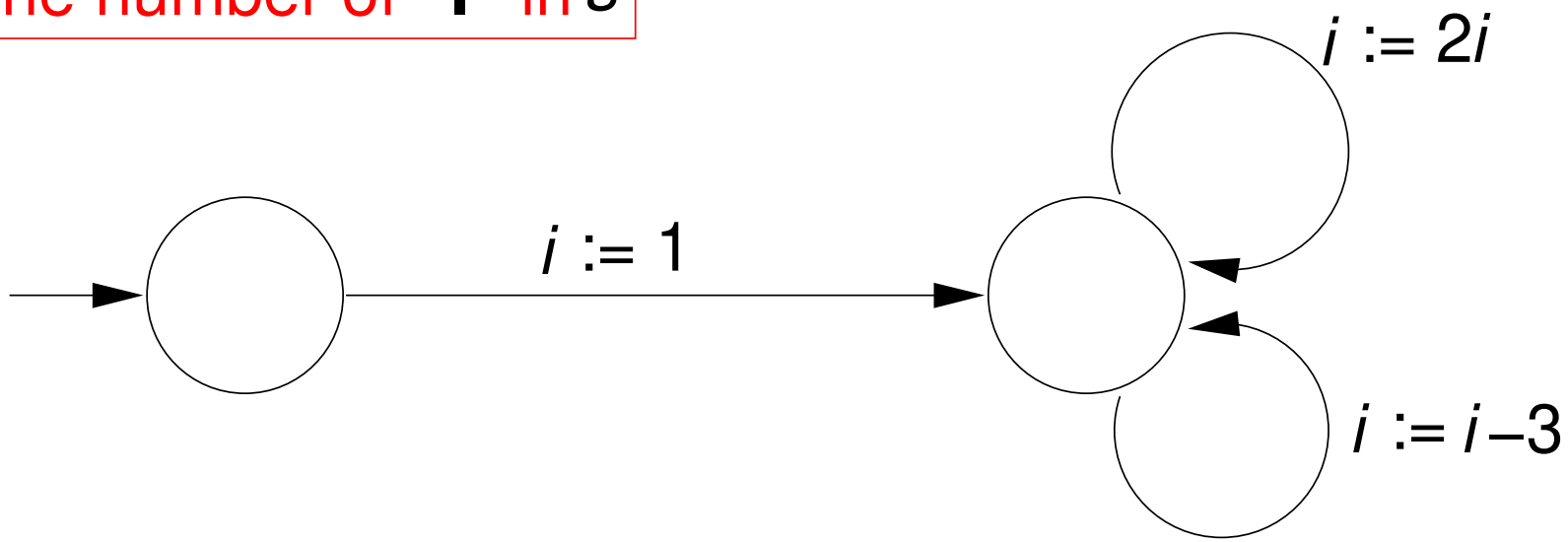
introduce "i" to count the number of "**I**" in s



Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \rightarrow s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \rightarrow s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \rightarrow s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \rightarrow s := \mathrm{XY}$;

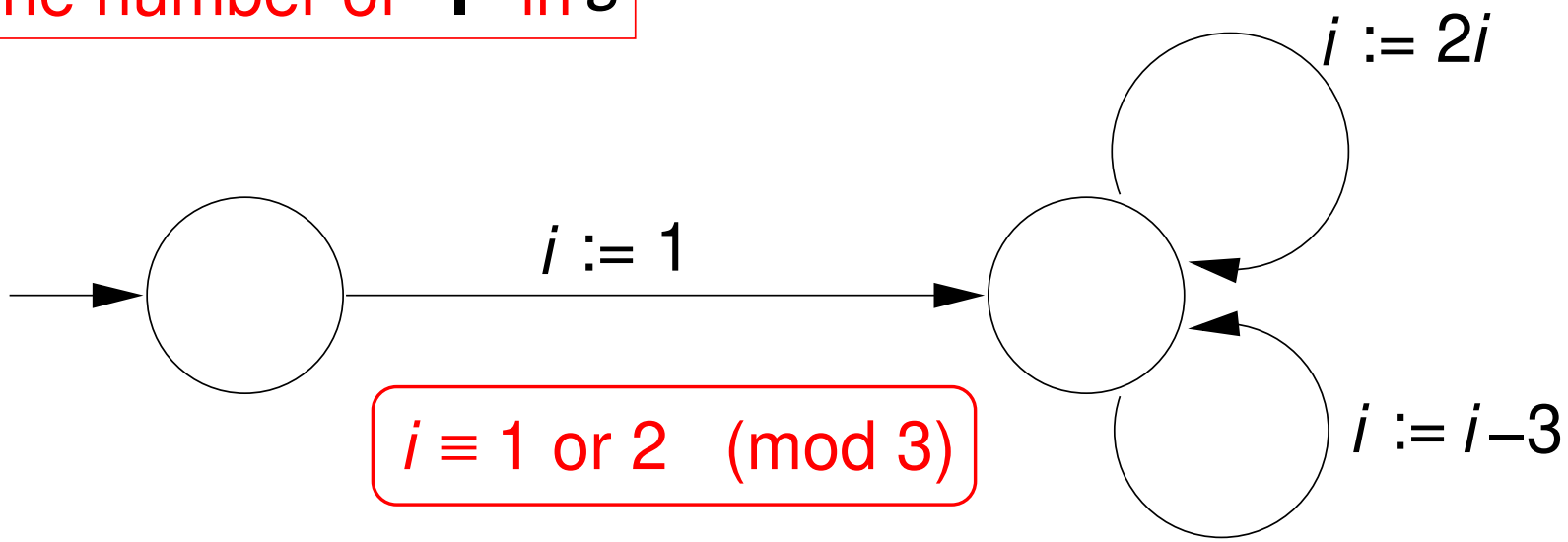
introduce "i" to count the number of "**I**" in s



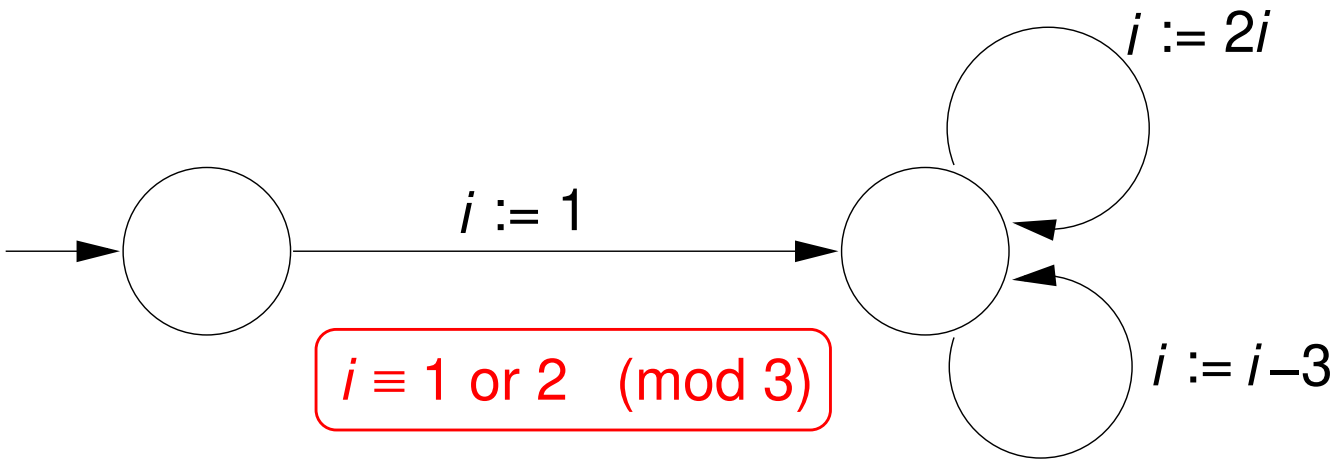
Can This (Nondeterministic) Program Halt?

$s := \mathbf{MI}$;
while $s \neq \mathbf{MU}$ do
choose
$\{s = \mathrm{X}\mathbf{I}\} \to s := \mathrm{X}\mathbf{IU}$;
$\{s = \mathbf{M}\mathrm{X}\} \to s := \mathbf{M}\mathrm{XX}$;
$\{s = \mathrm{X}\mathbf{III}\mathrm{Y}\} \to s := \mathrm{X}\mathbf{U}\mathrm{Y}$;
$\{s = \mathrm{X}\mathbf{UU}\mathrm{Y}\} \to s := \mathrm{XY}$;

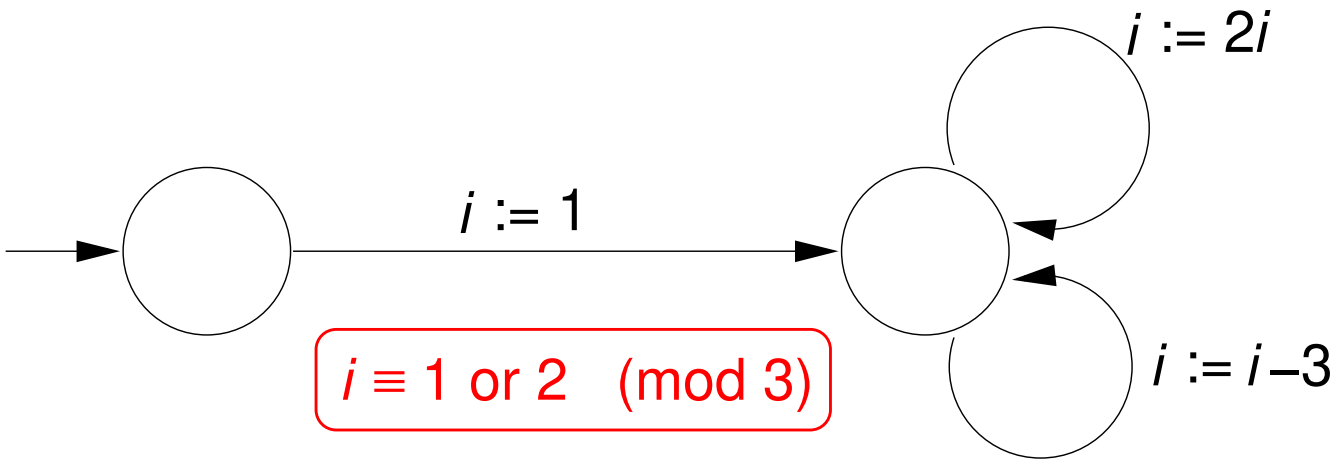
introduce "i" to count the number of "**I**" in s



A Polynomial Program Invariant

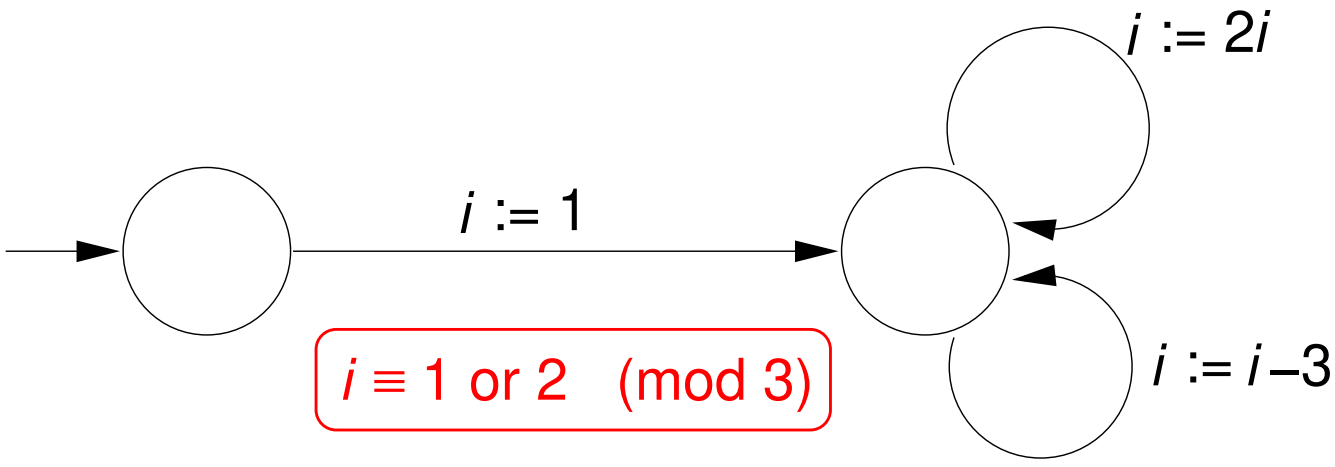


A Polynomial Program Invariant



- Define the polynomial $p(x) = (x-1)(x-2)$

A Polynomial Program Invariant



- Define the polynomial $p(x) = (x-1)(x-2)$
- Then this program satisfies

$$p(i) = 0$$

(over $\mathbb{Z}_3$)

Automated Invariant Generation: A 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

Inductive Invariants

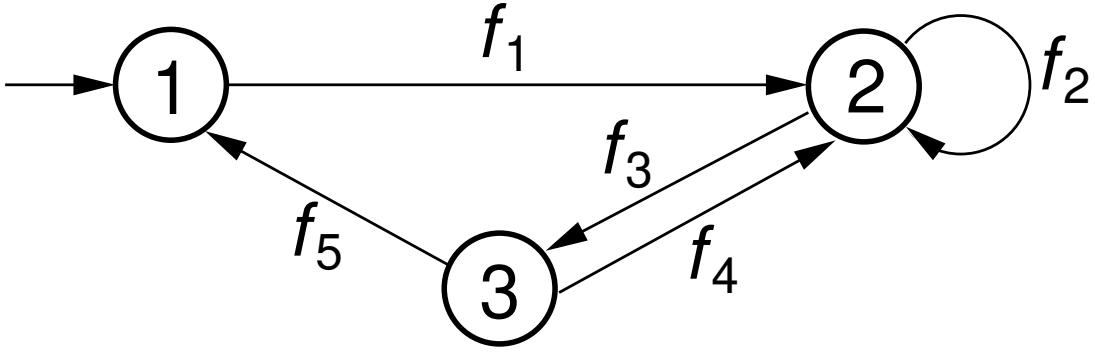
invariant = overapproximation (of the reachable states)

Inductive Invariants

invariant = overapproximation (of the reachable states)

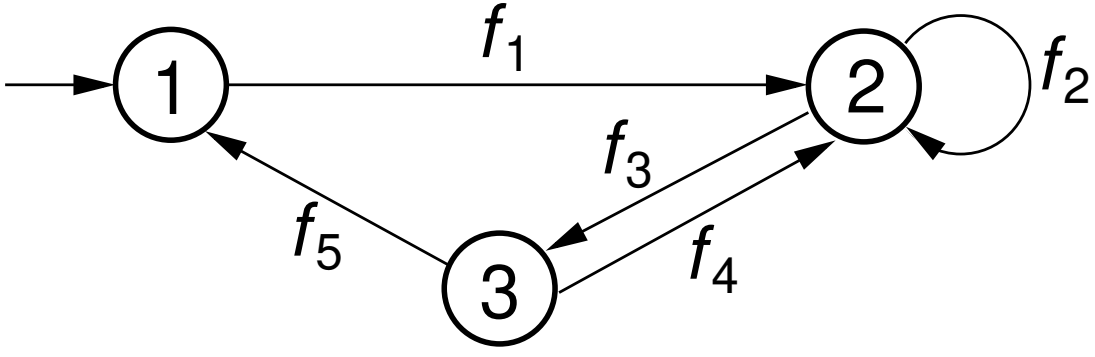
$$\textbf{inductive invariant} = \begin{cases} \text{overapproximation} \\ \text{preserved by the transition relation} \end{cases}$$

Inductive Invariants



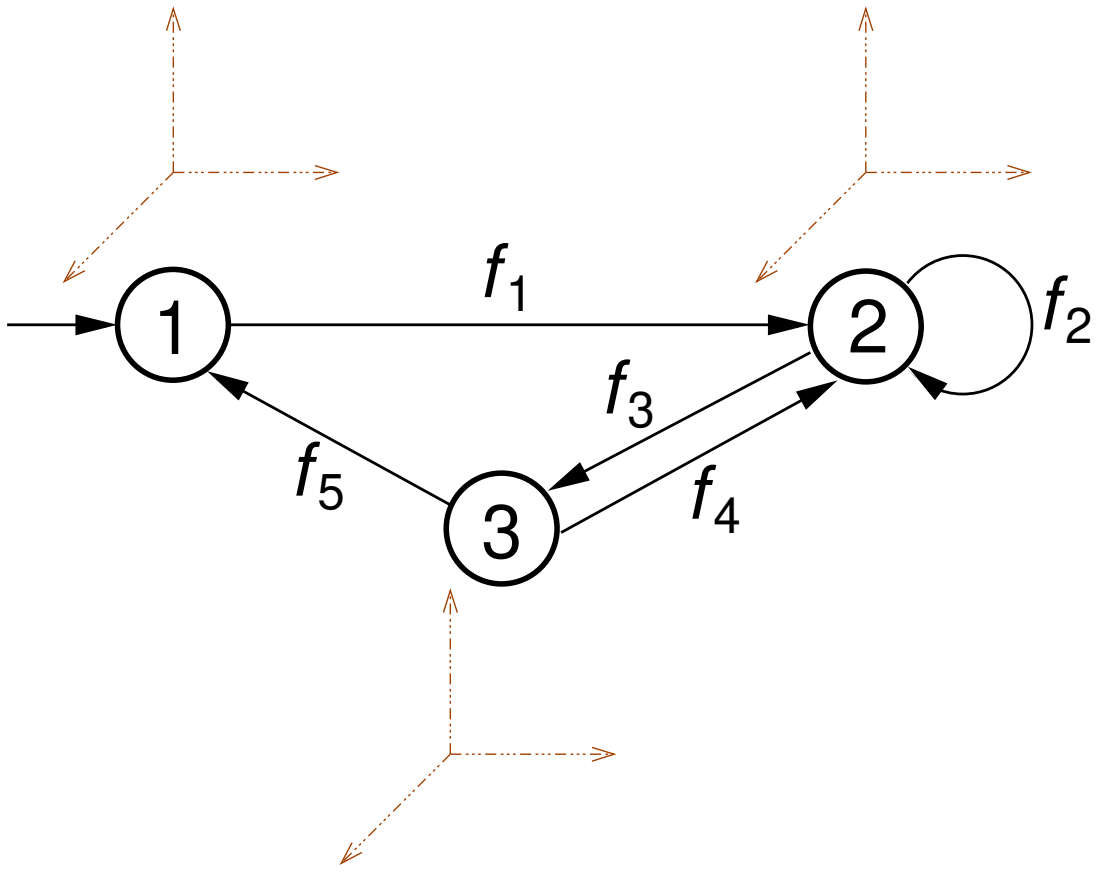
Inductive Invariants

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



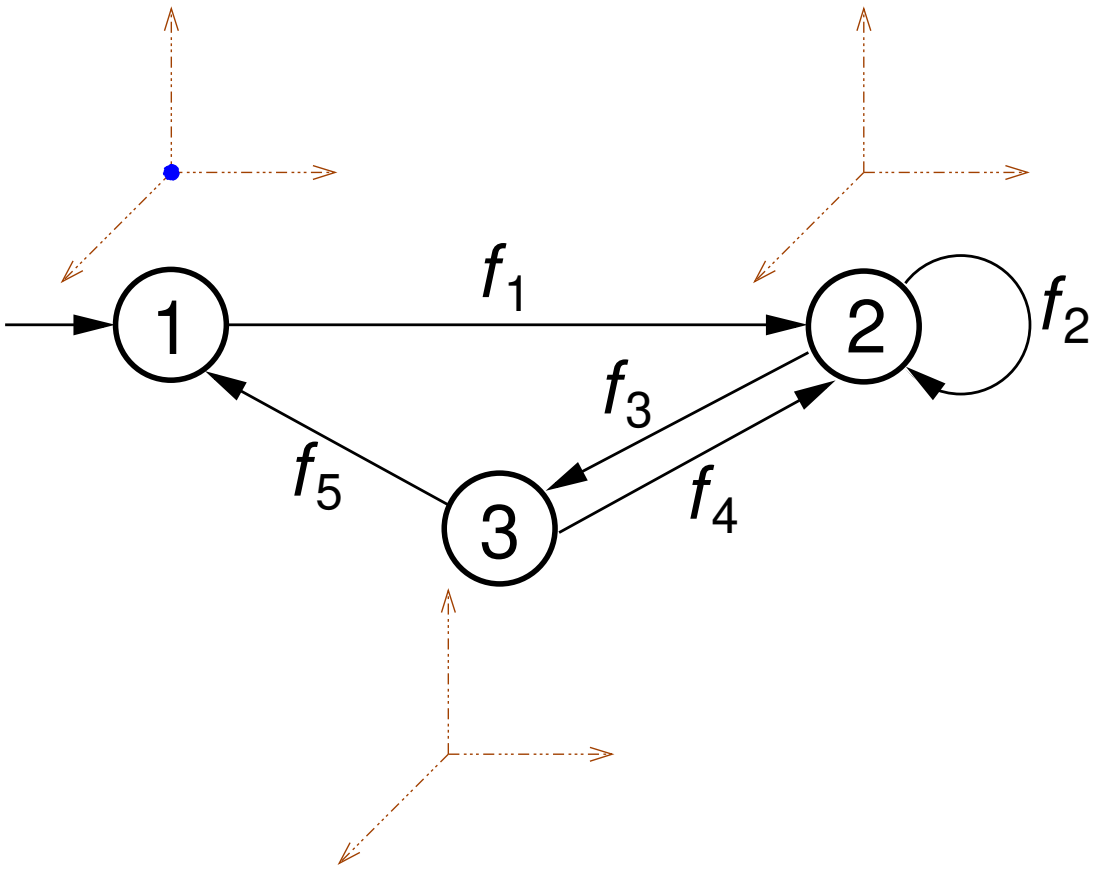
Inductive Invariants

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



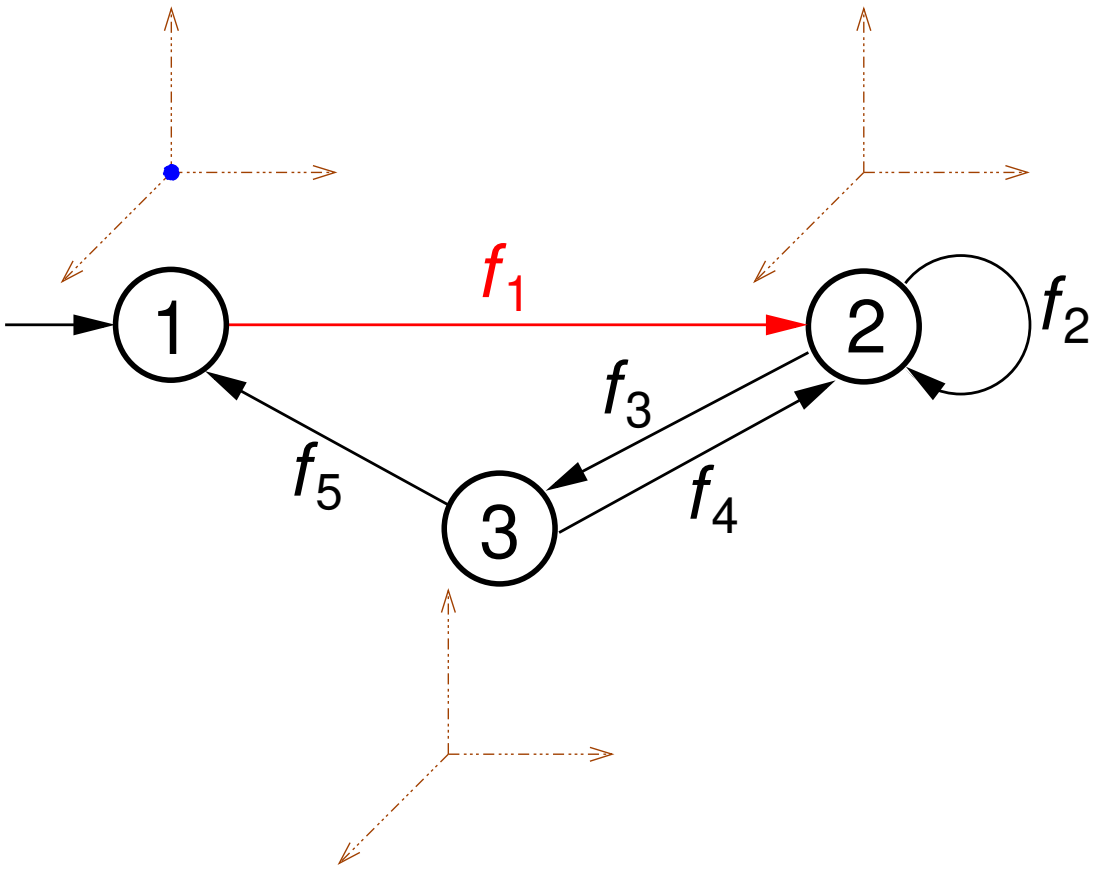
Inductive Invariants

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



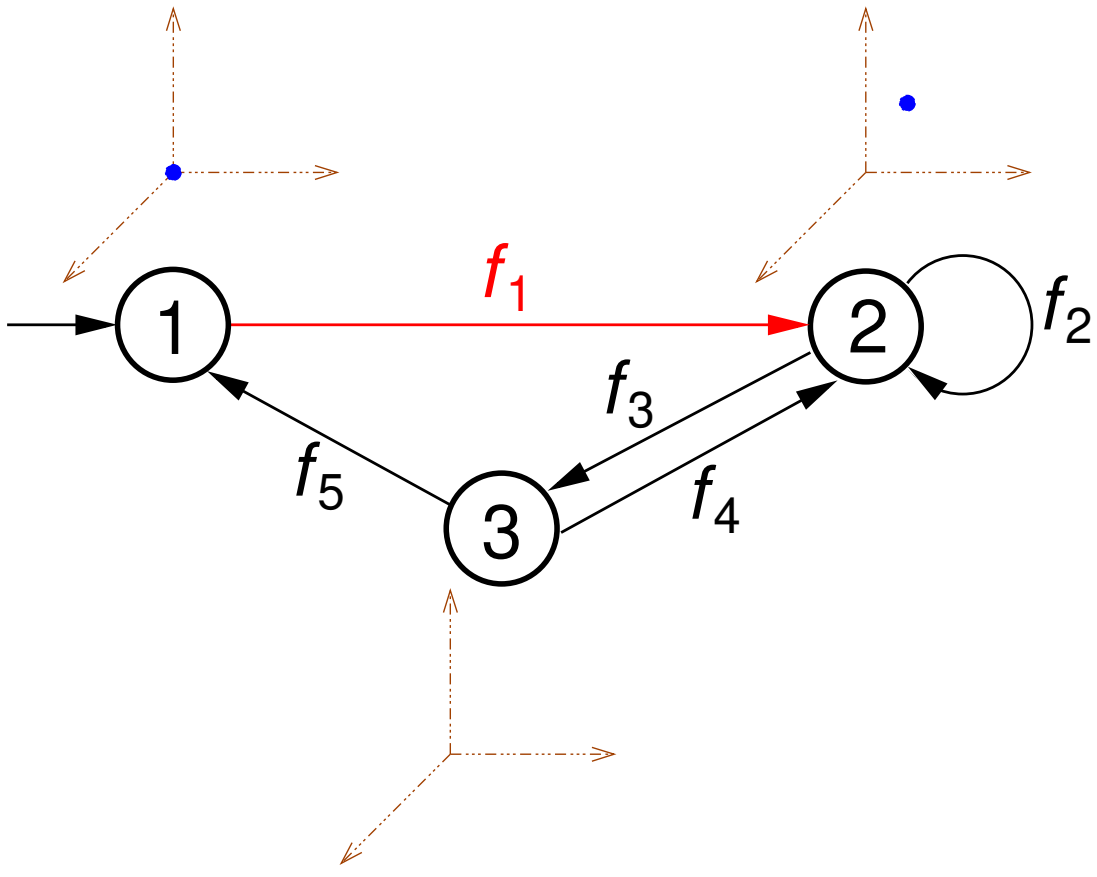
Inductive Invariants

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



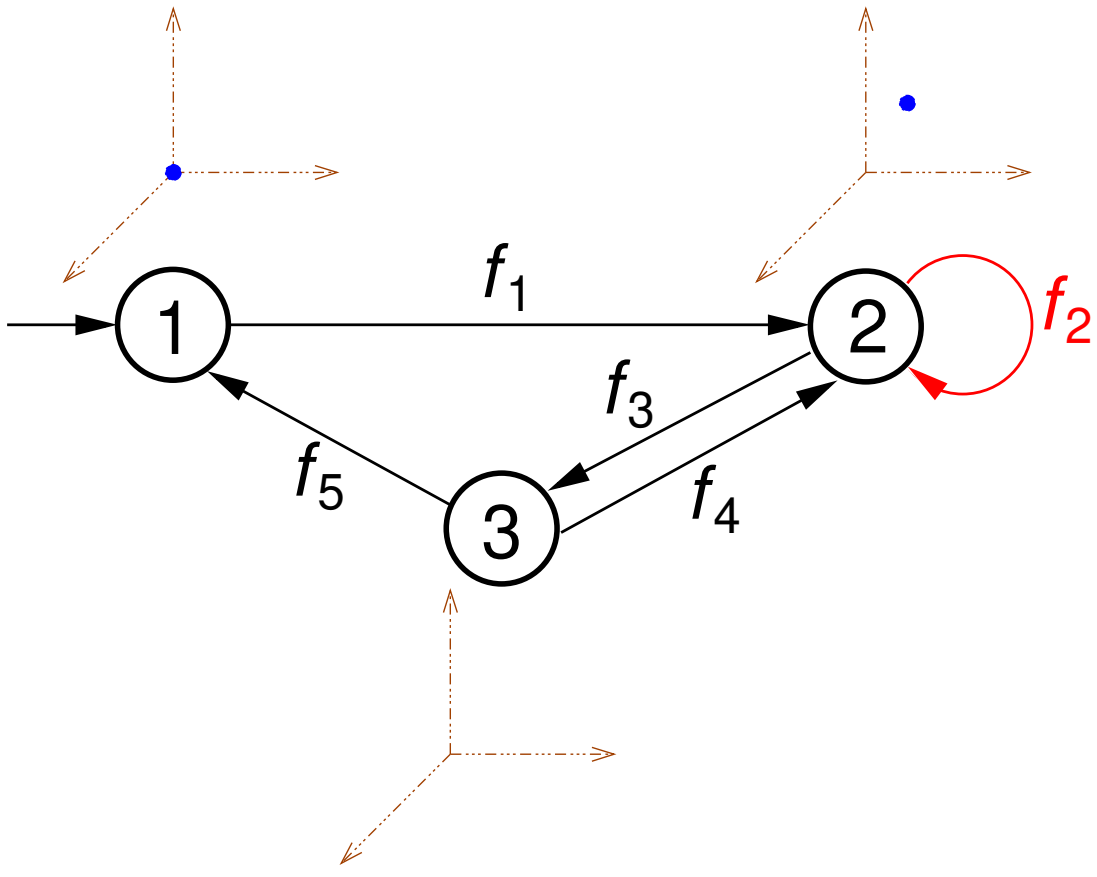
Inductive Invariants

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



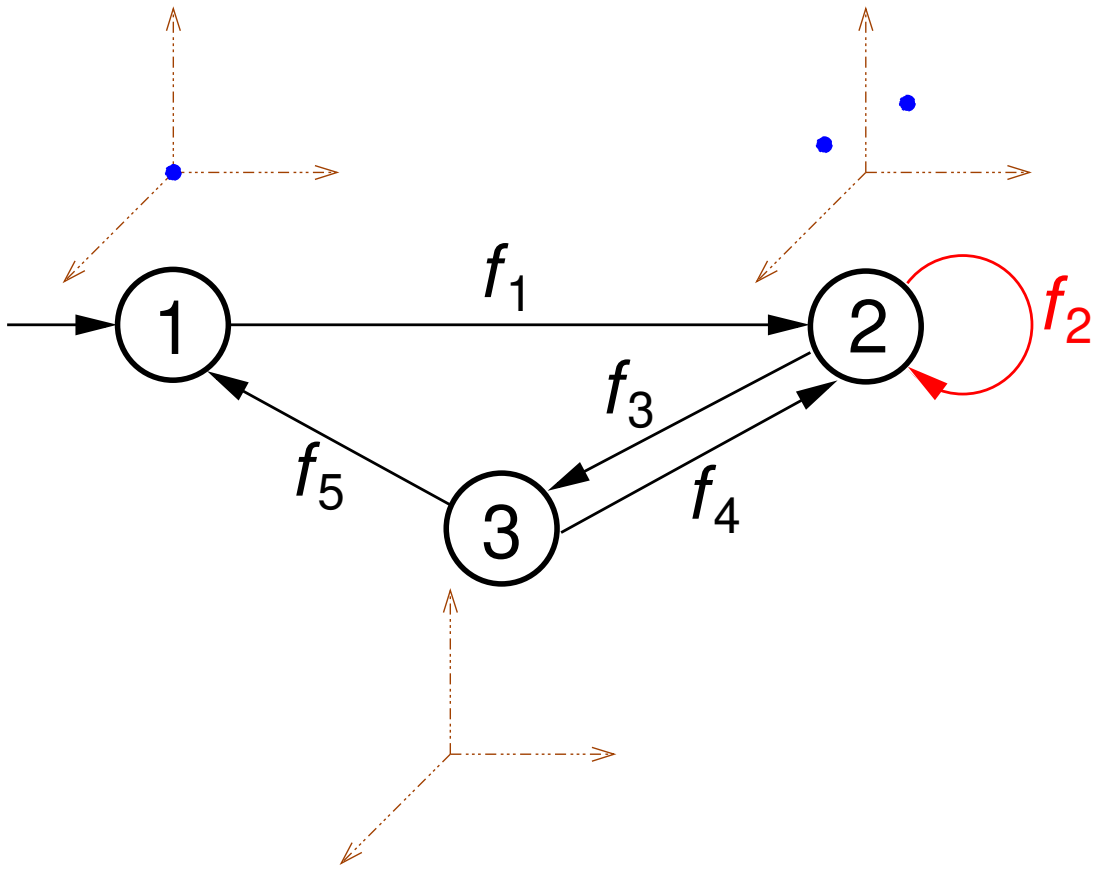
Inductive Invariants

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



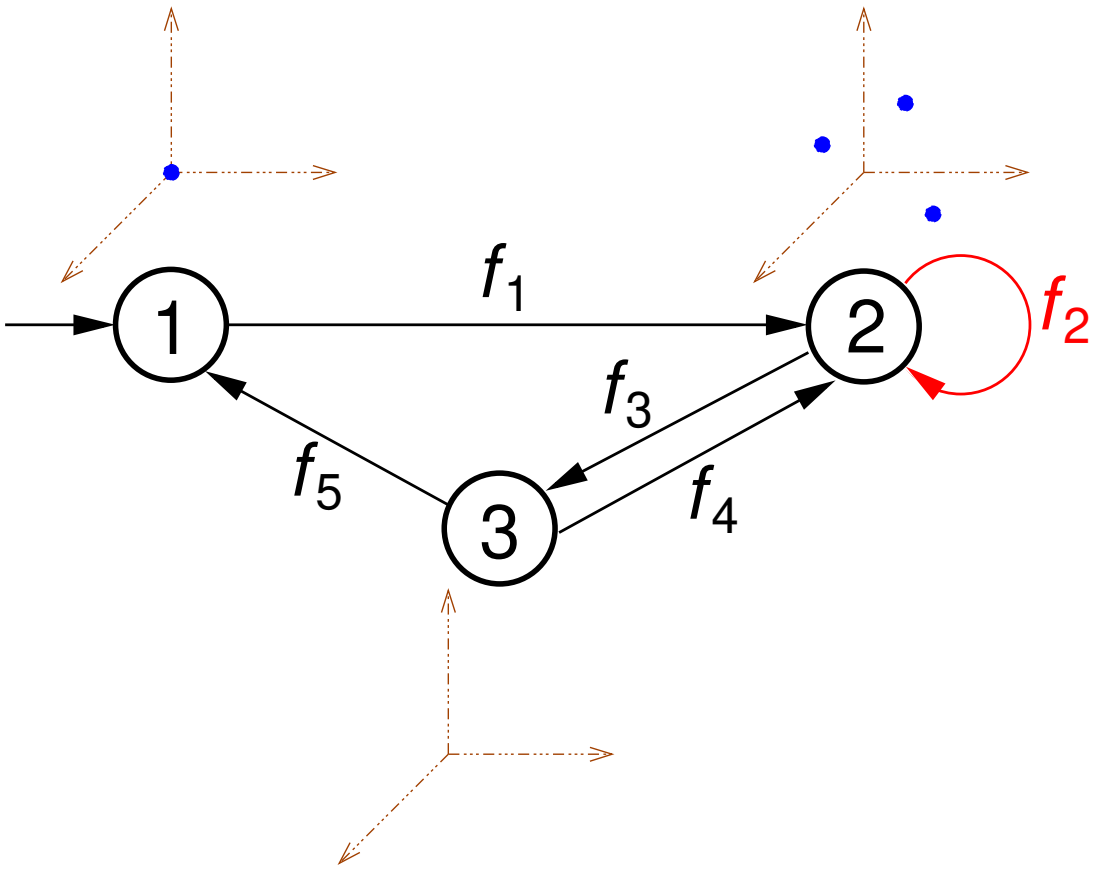
Inductive Invariants

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



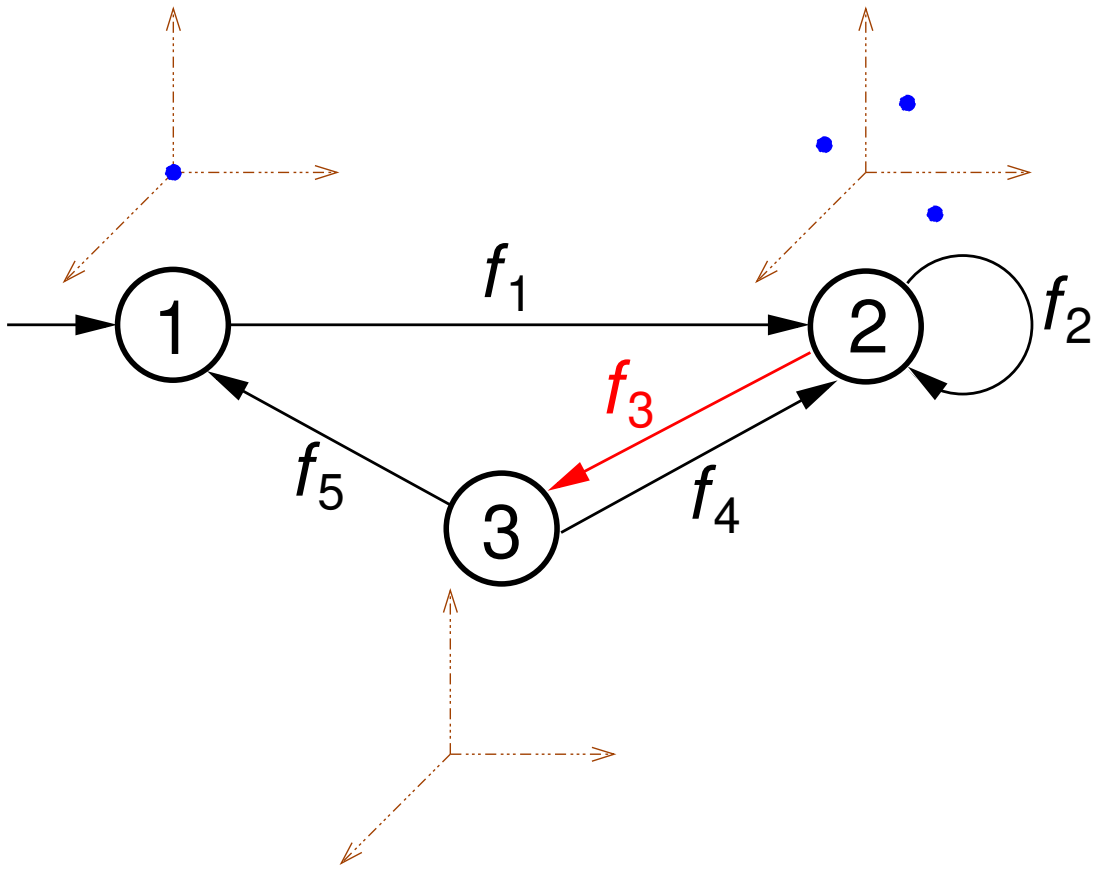
Inductive Invariants

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



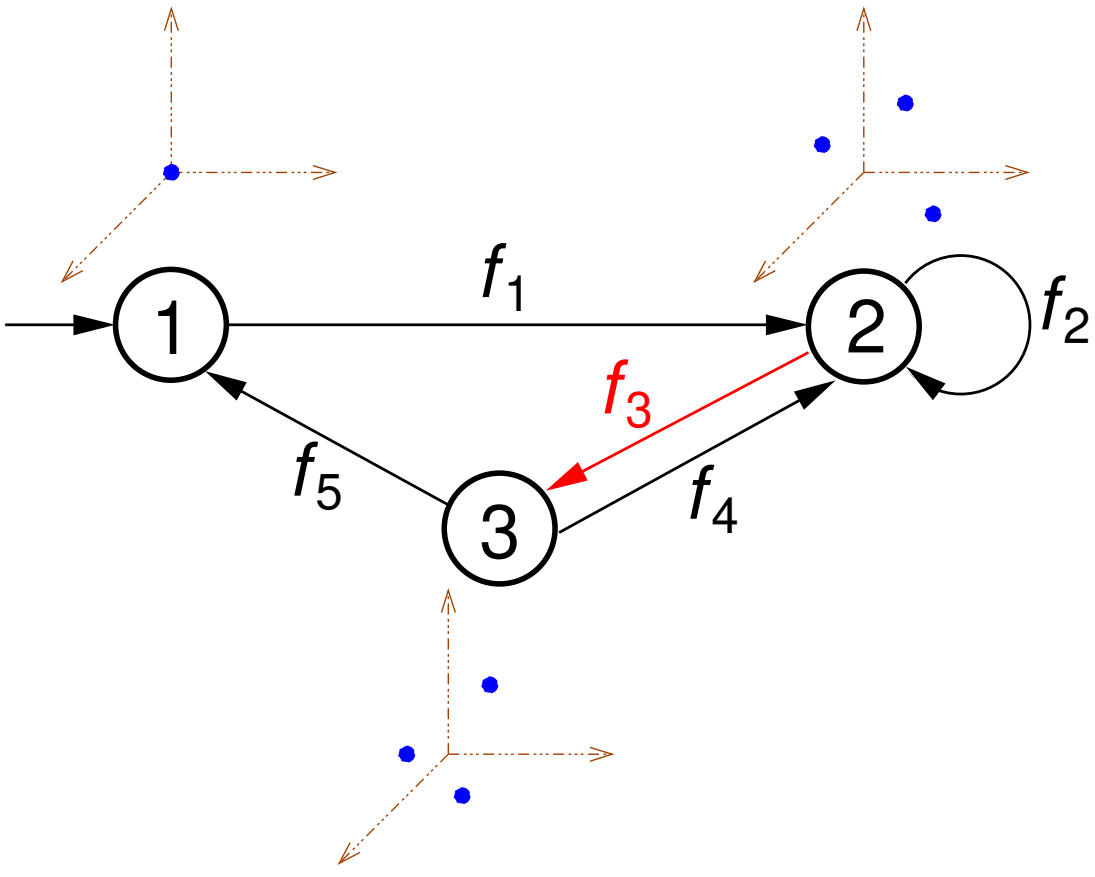
Inductive Invariants

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



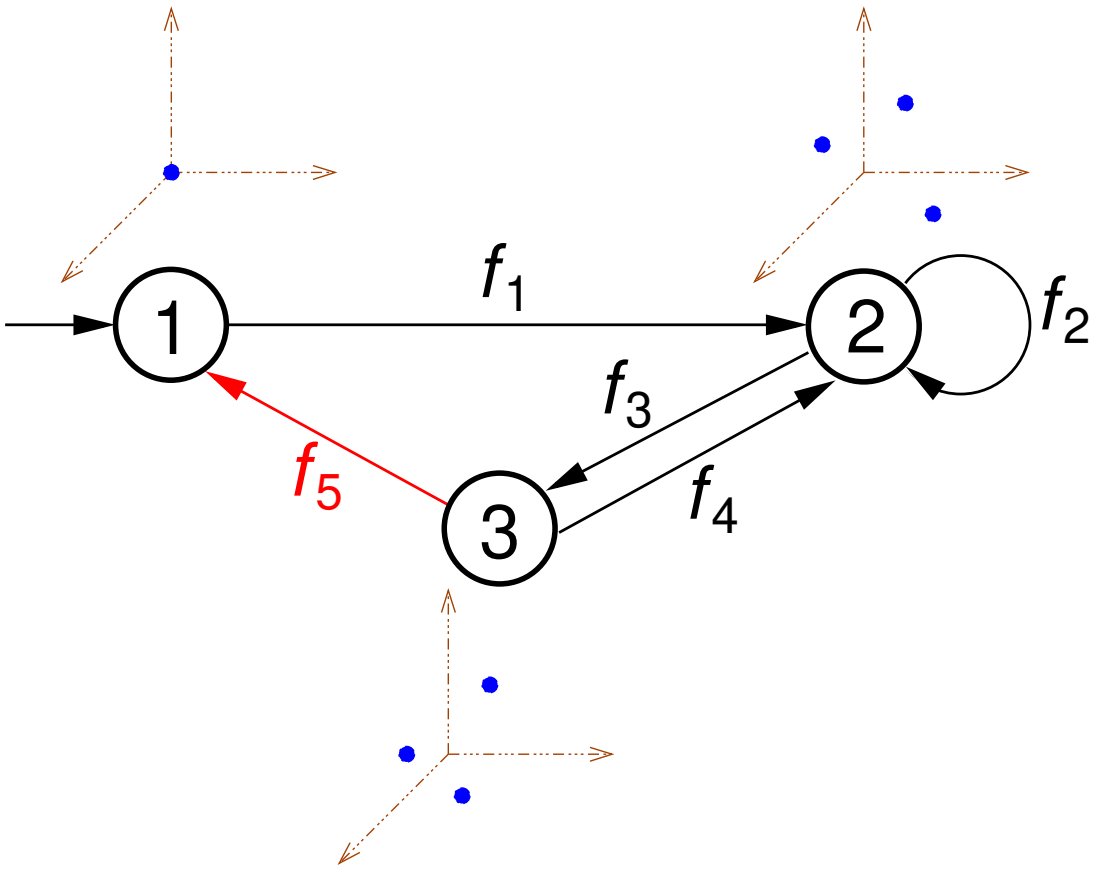
Inductive Invariants

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



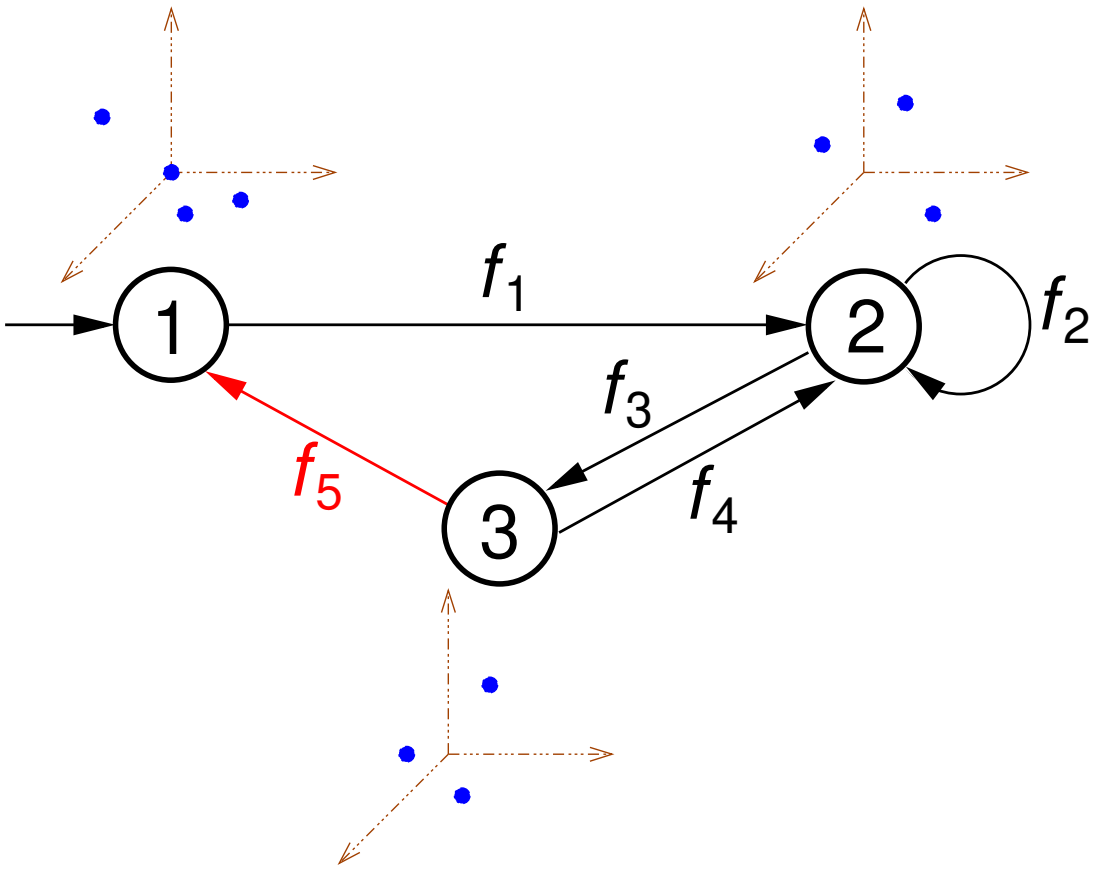
Inductive Invariants

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



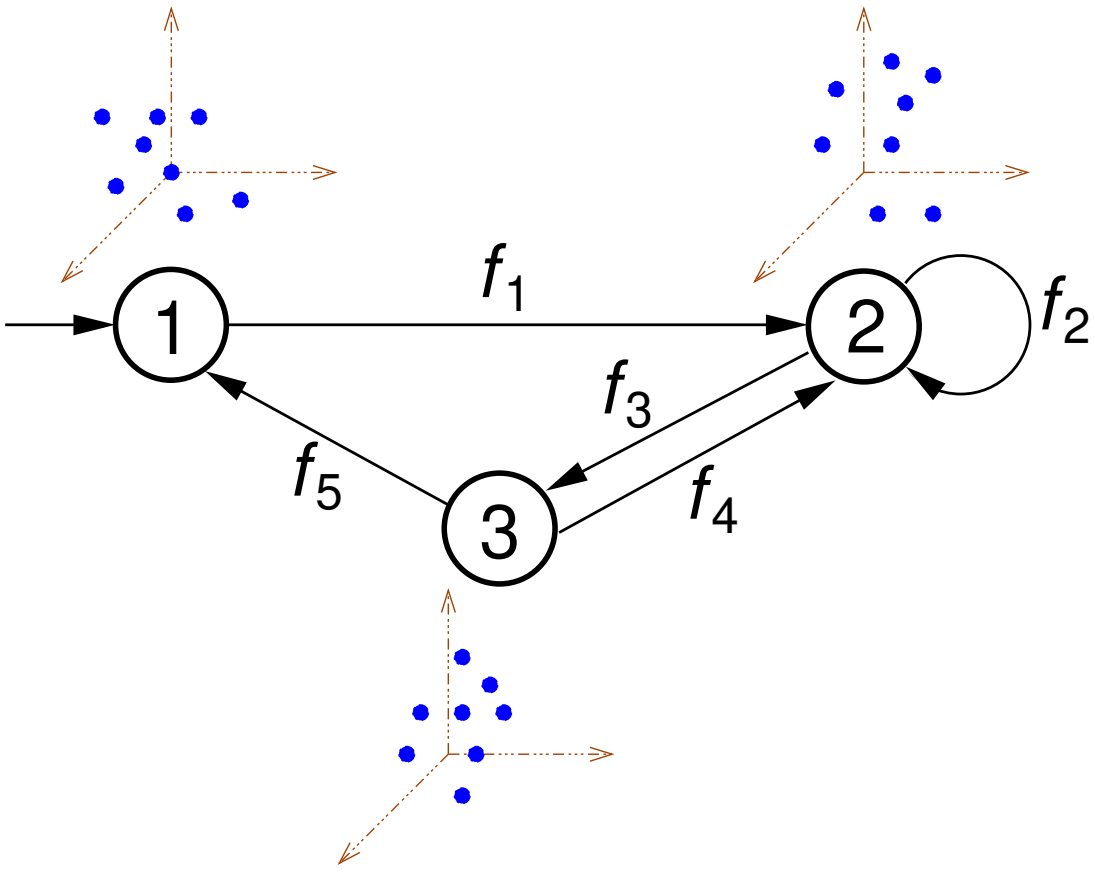
Inductive Invariants

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



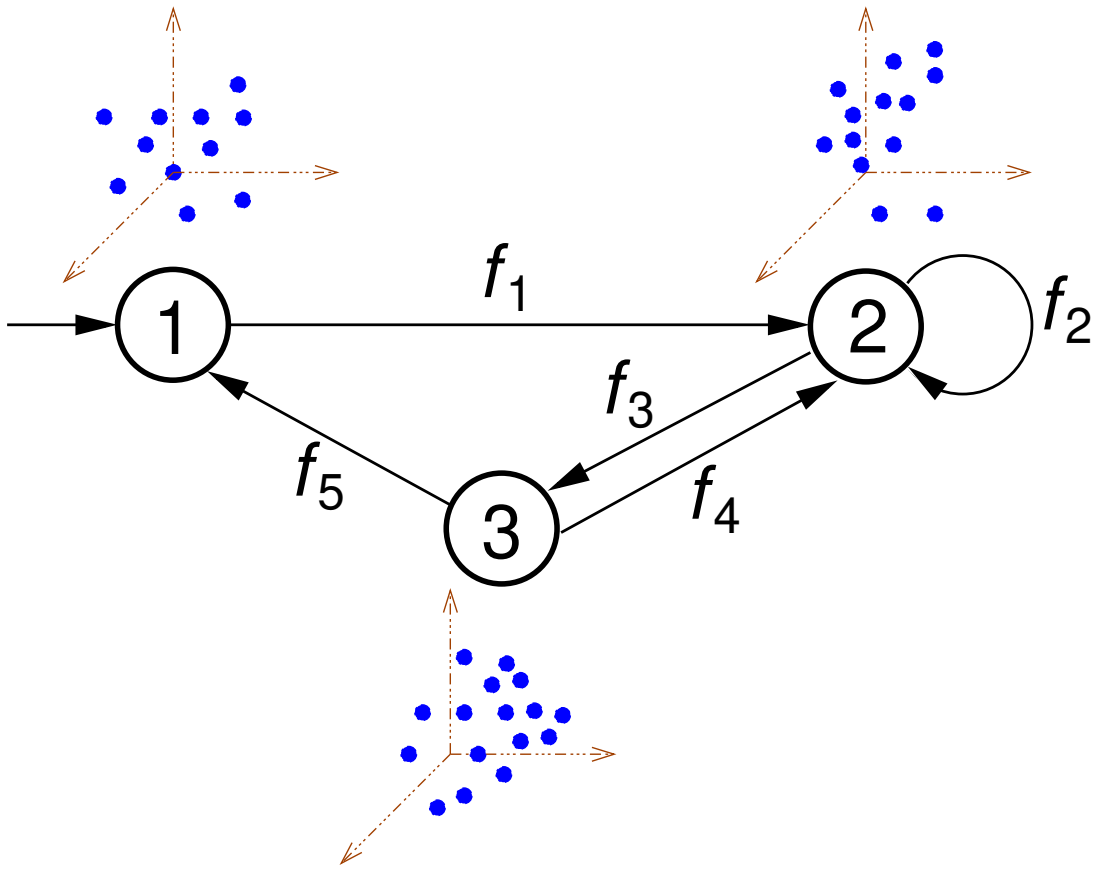
Inductive Invariants

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



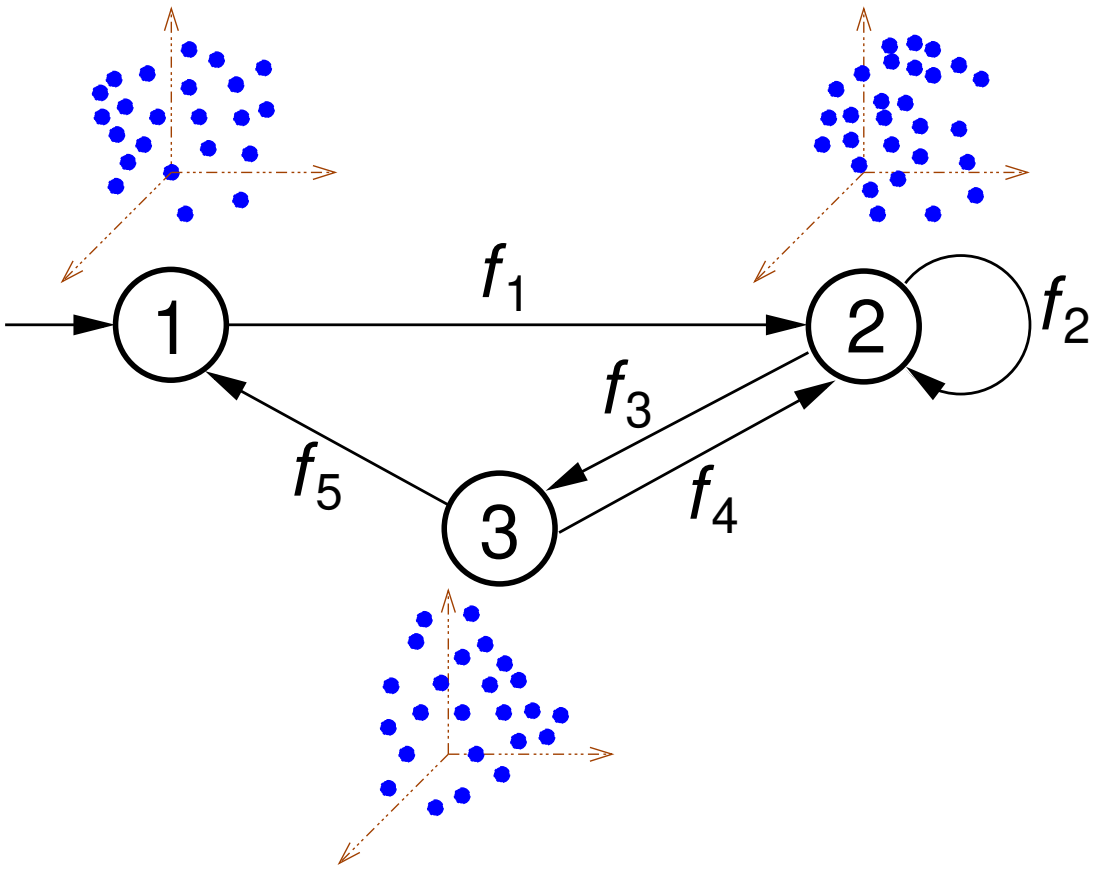
Inductive Invariants

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



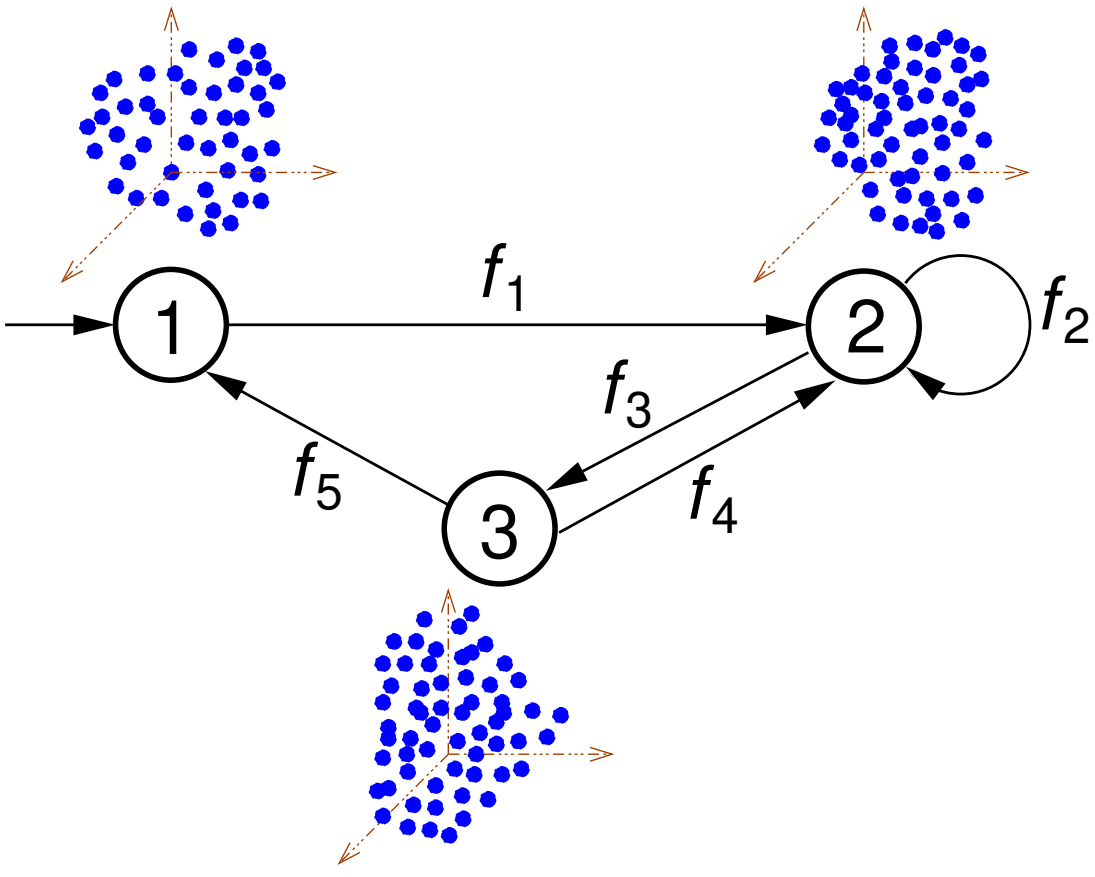
Inductive Invariants

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



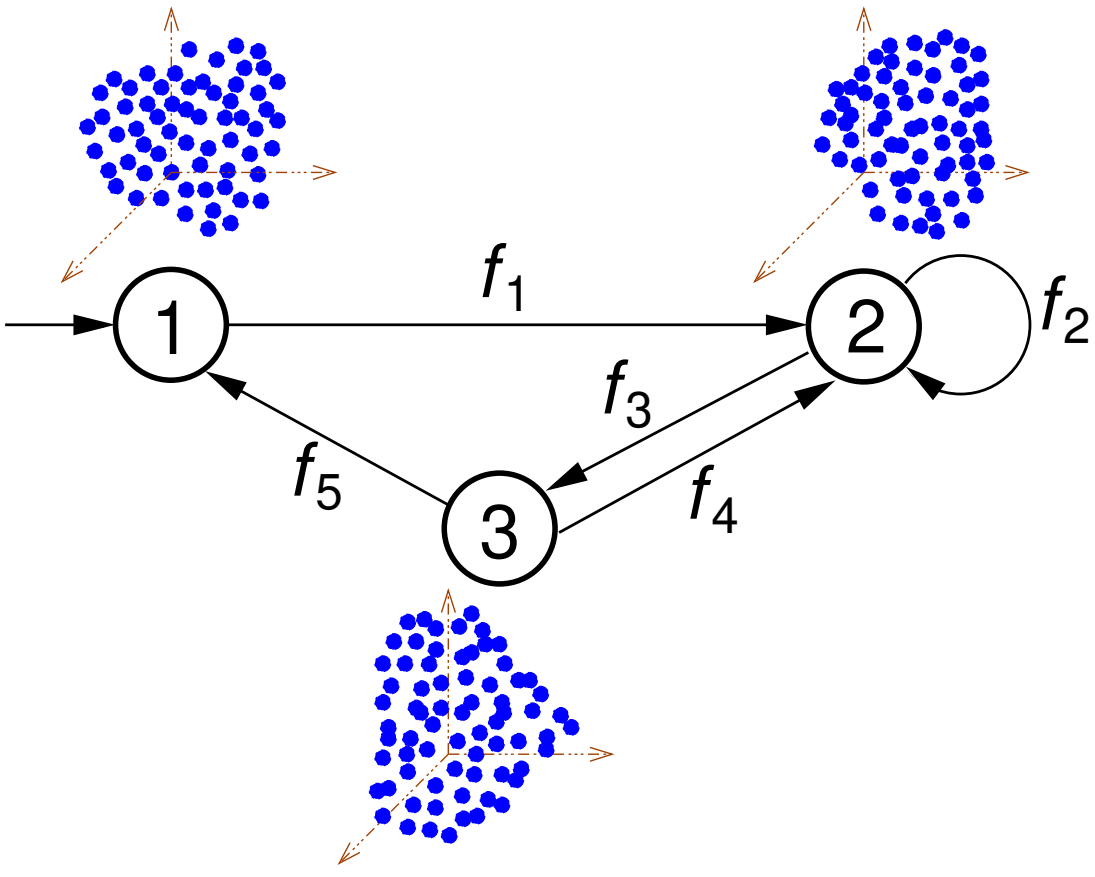
Inductive Invariants

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



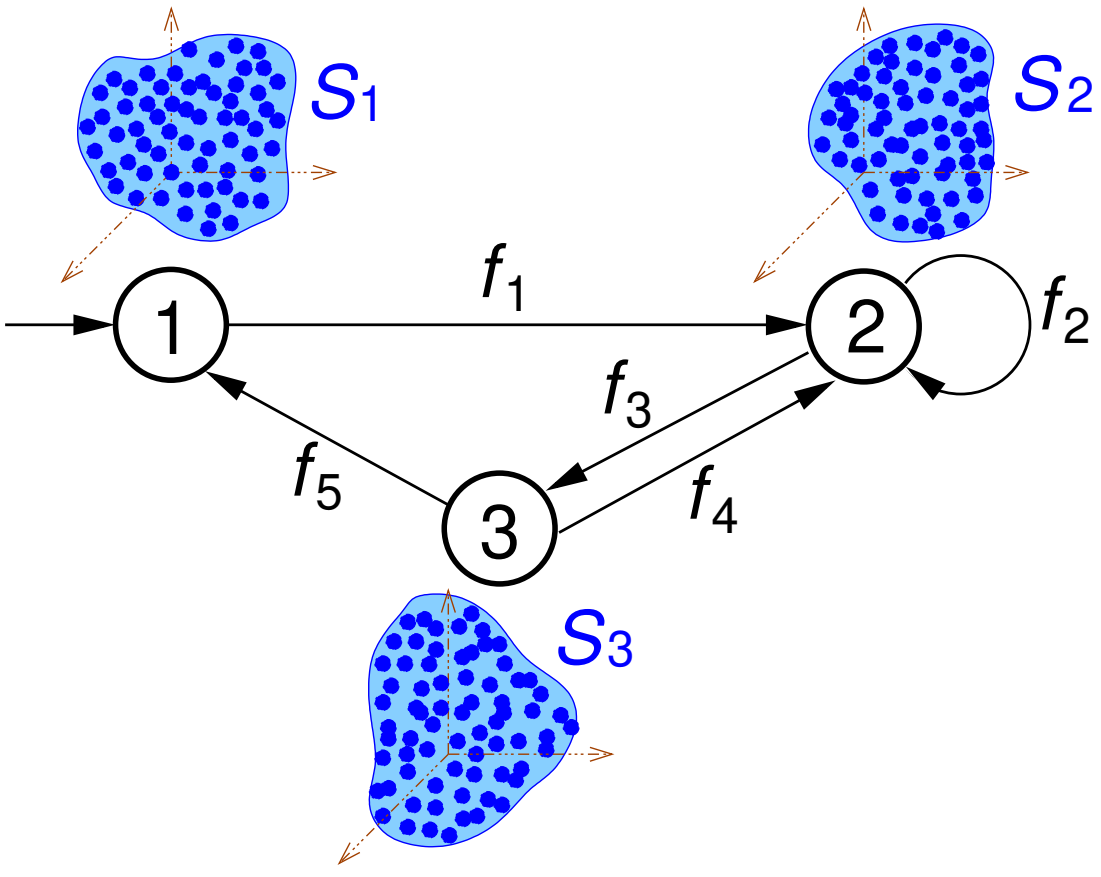
Inductive Invariants

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



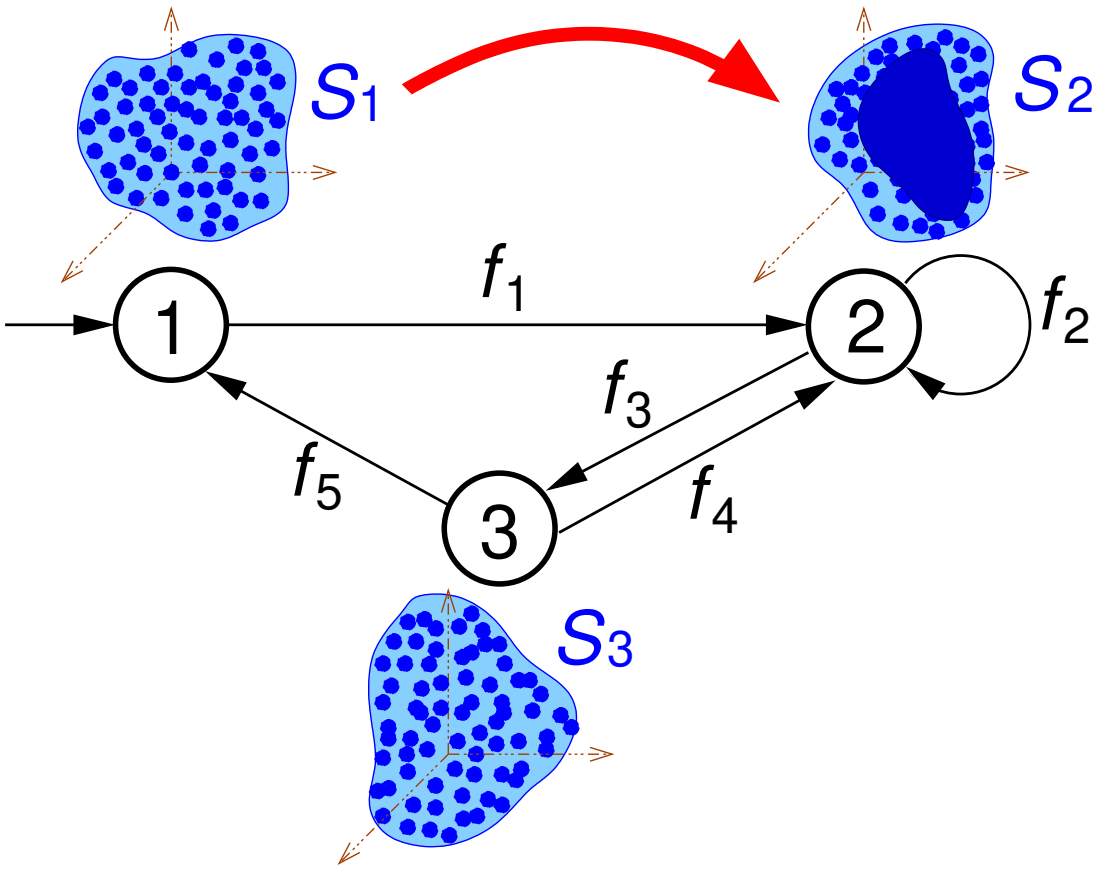
Inductive Invariants

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



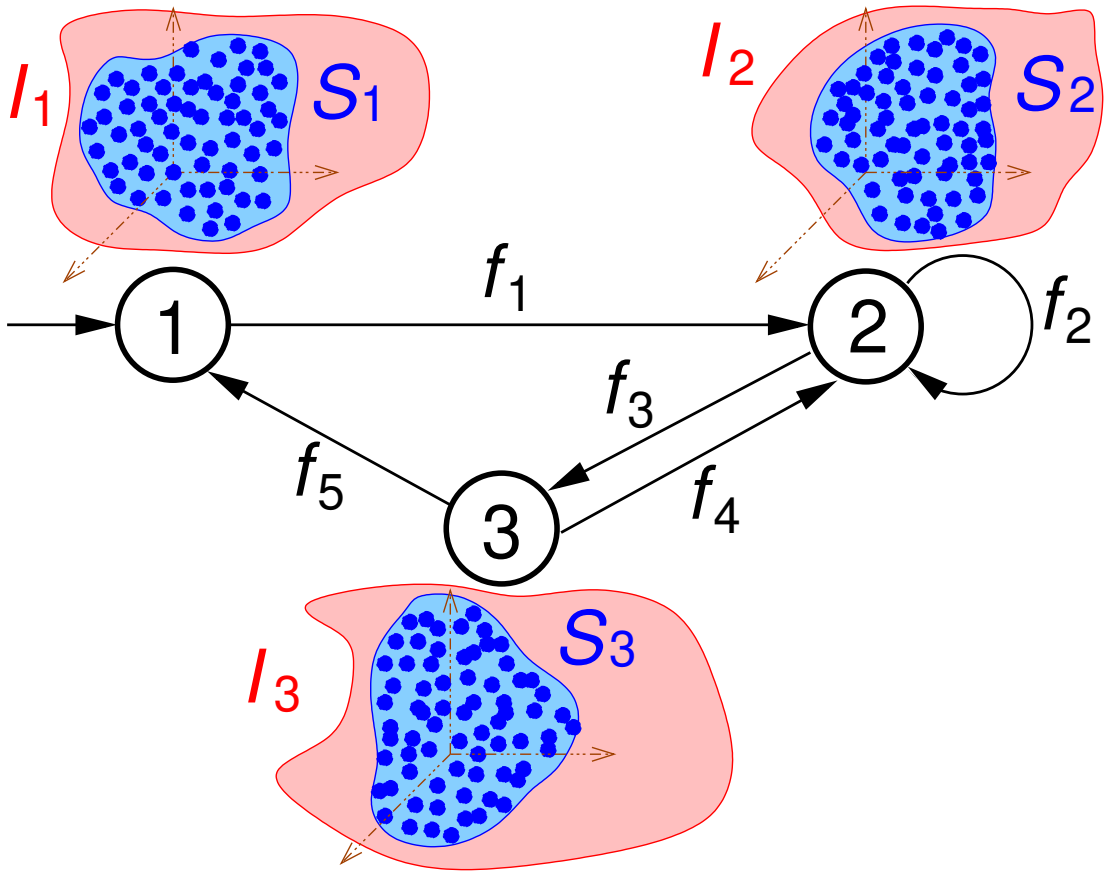
Inductive Invariants

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



Inductive Invariants

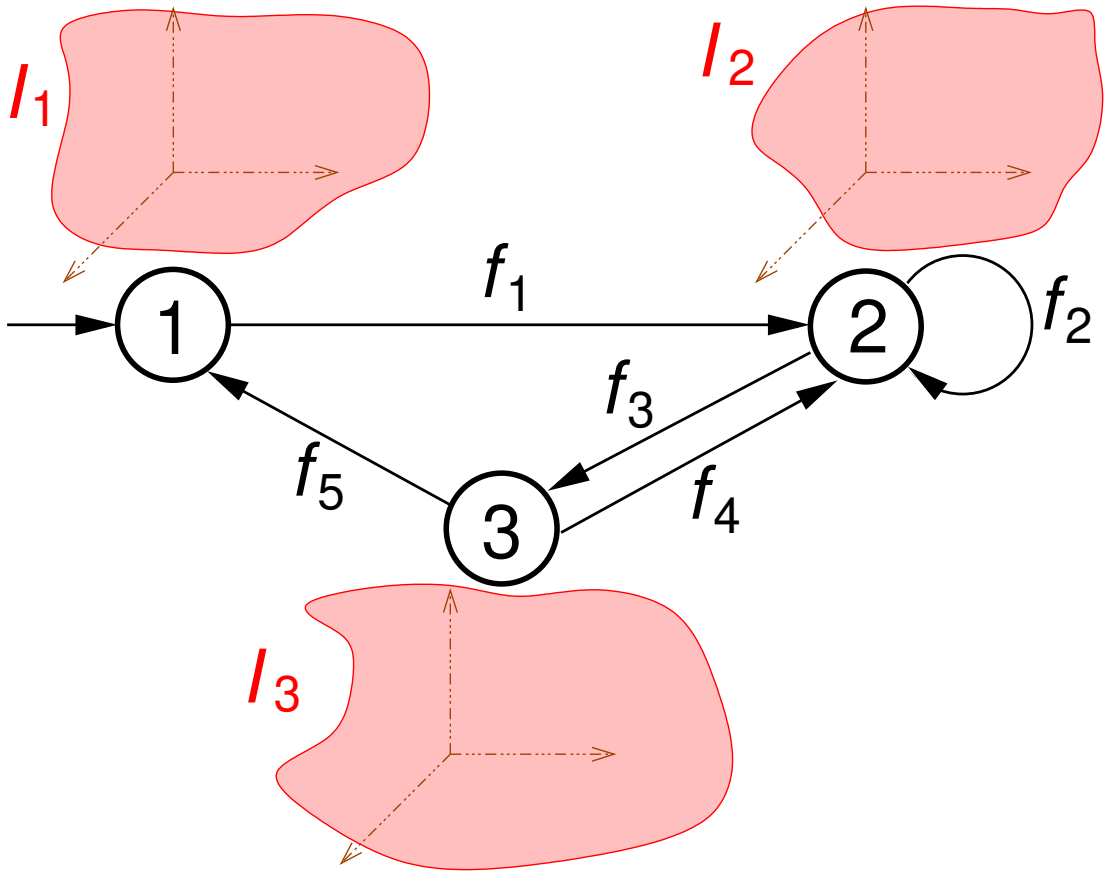
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

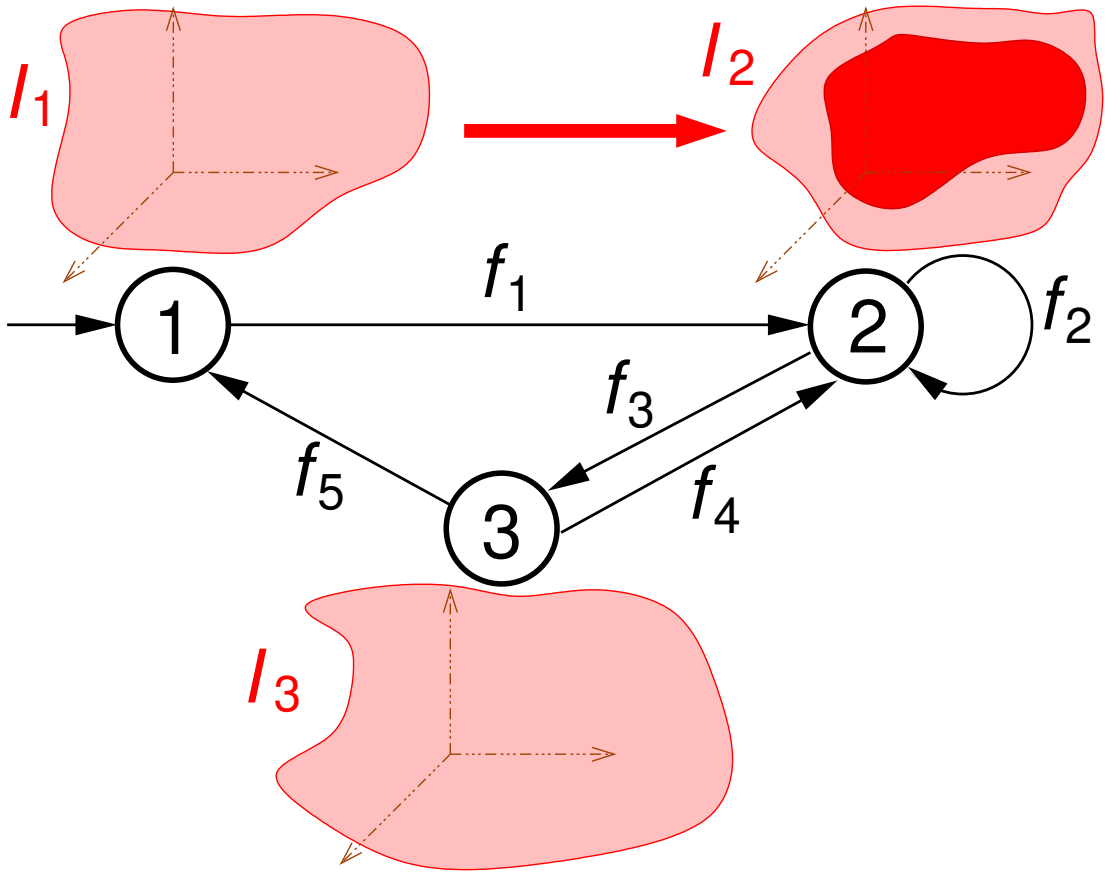
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

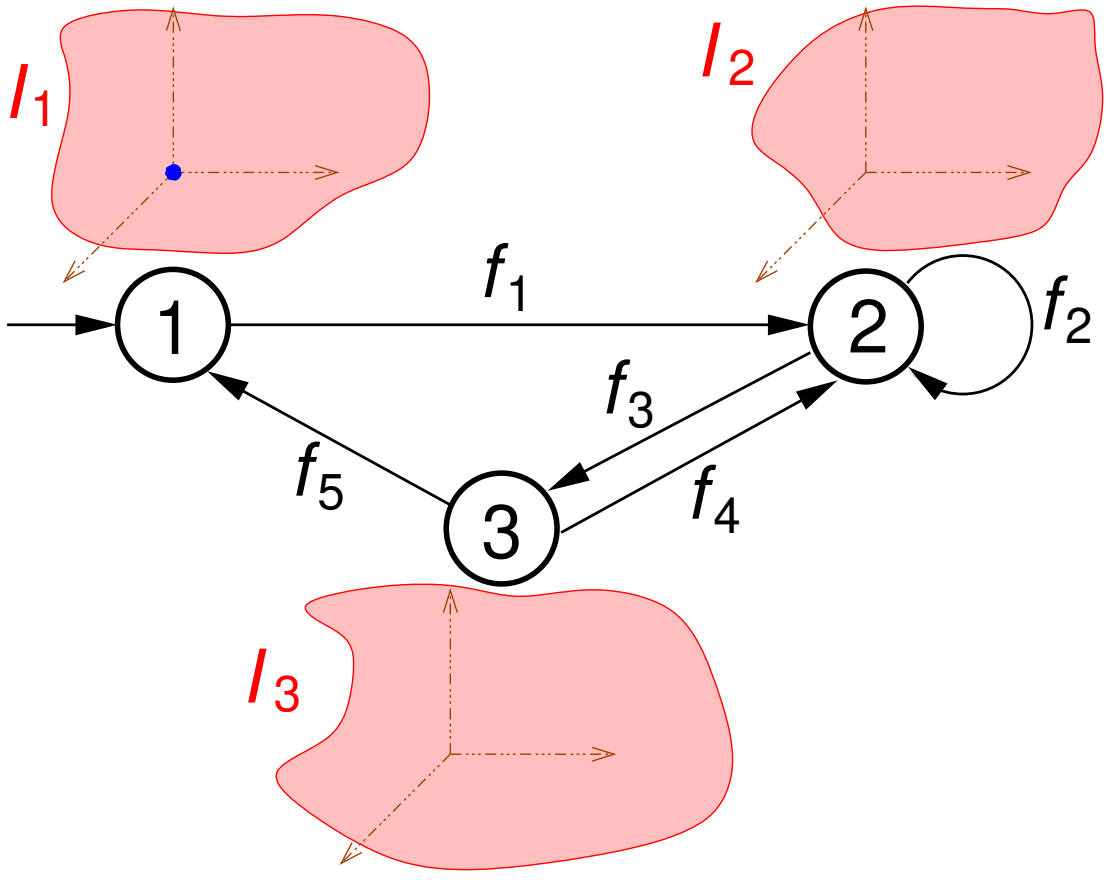
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

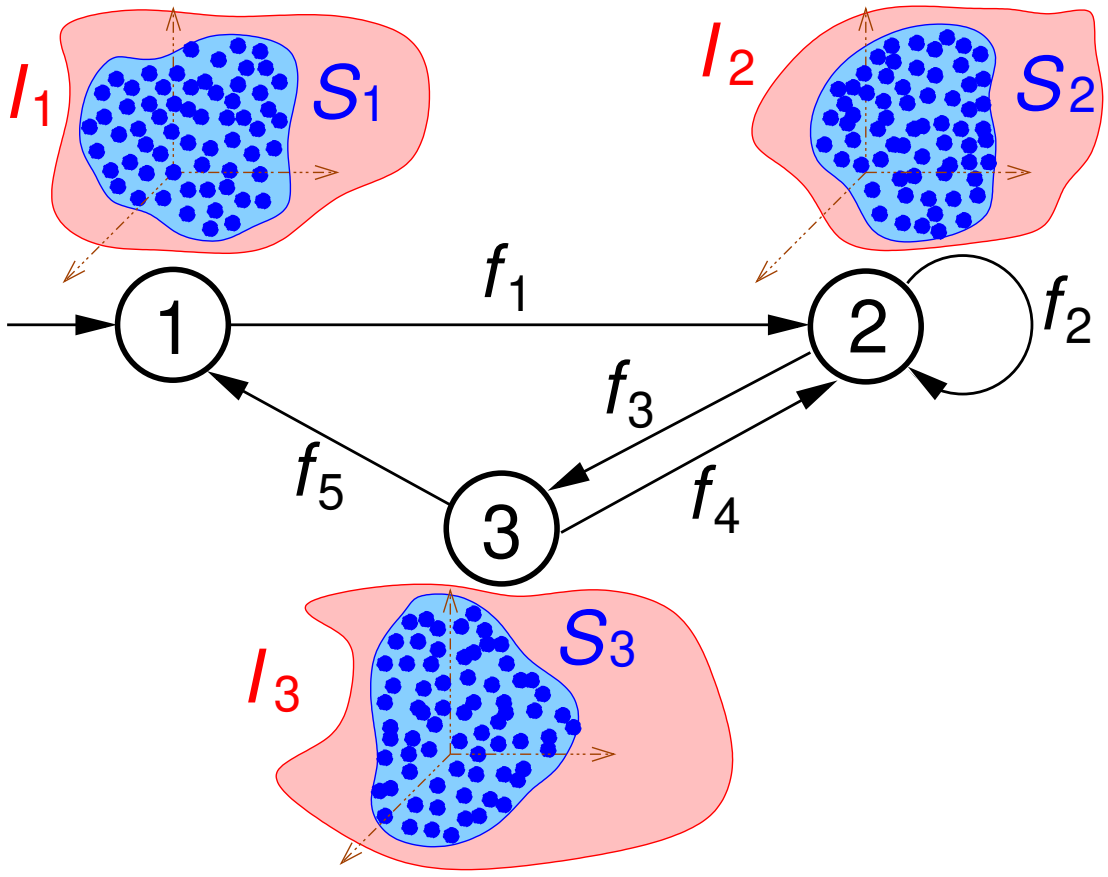
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

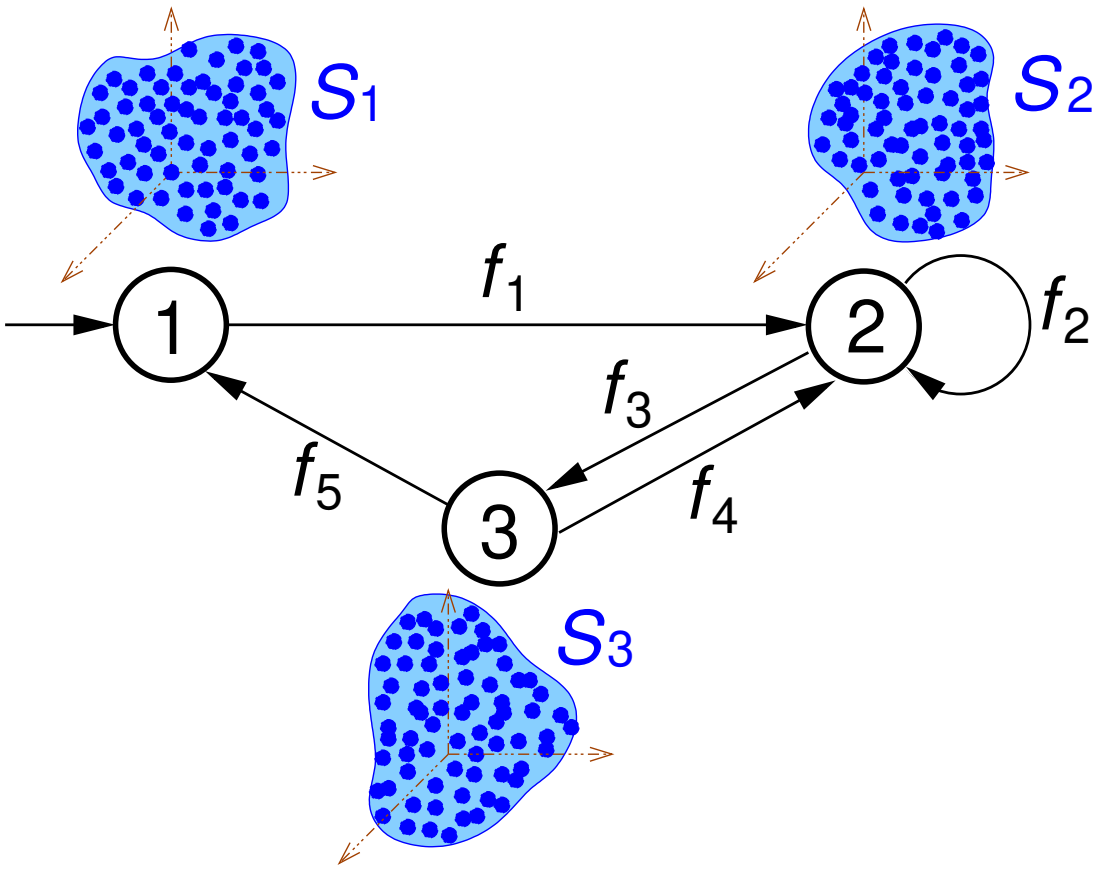
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

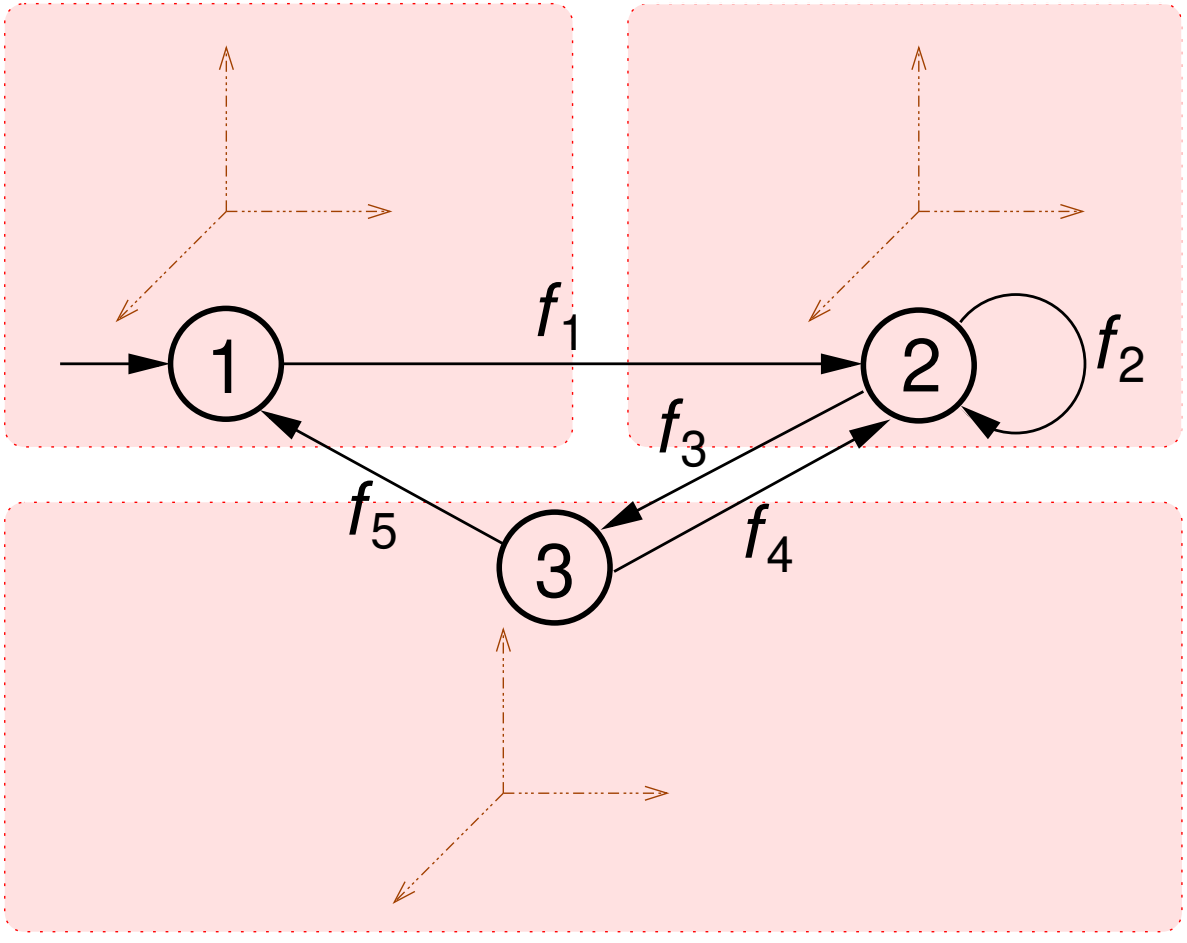
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



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

Inductive Invariants

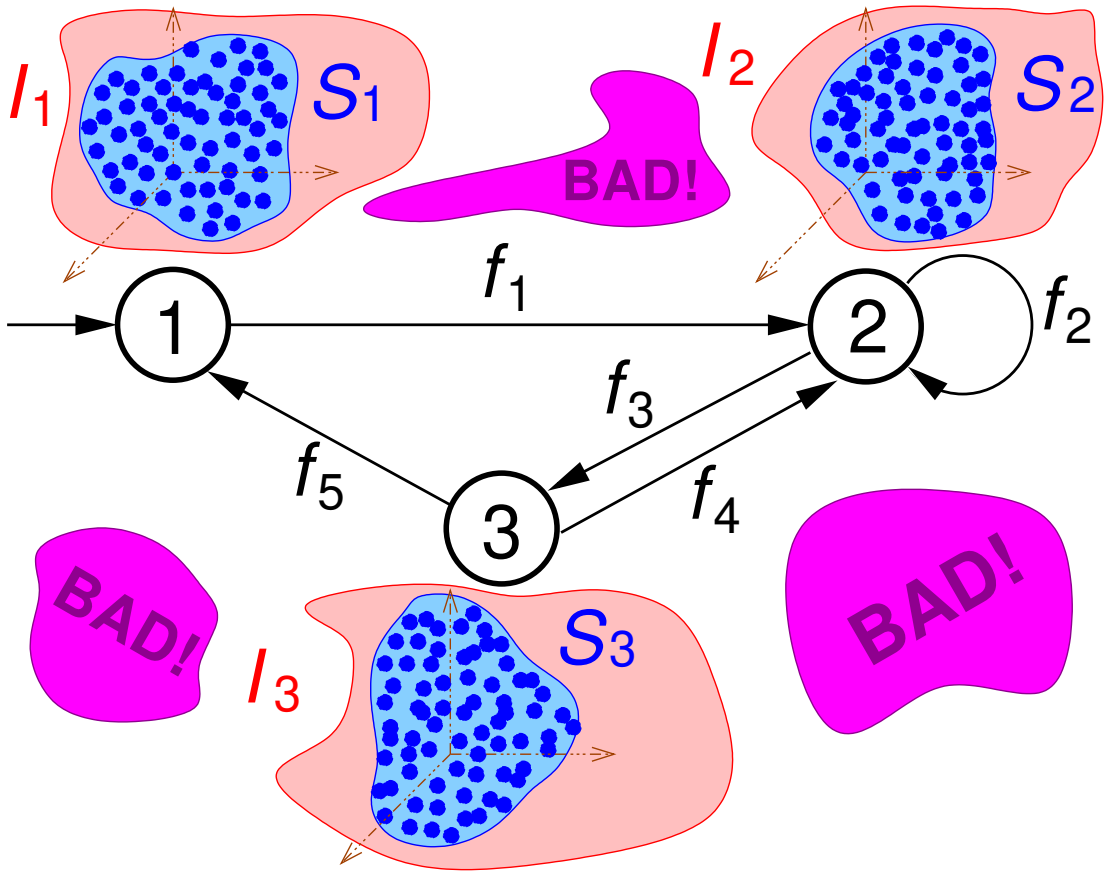
x, y, z range over $\mathbb{Z}$ (or $\mathbb{Q}$)



$\langle \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3 \rangle$ 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
  - Often trade-off between precision and complexity . . .

A Menagerie of Abstract Domains

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

A Menagerie of Abstract Domains

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

A Menagerie of Abstract Domains

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

A Menagerie of Abstract Domains

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

- **Polyhedral / Semilinear sets** [Cousot, Halbwachs 78]

$$x + 2y - 3z + 4 \leq 0 \quad \vee \quad 2x + 7y + 2z \geq 0$$

A Menagerie of Abstract Domains

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

- **Polyhedral / Semilinear sets** [Cousot, Halbwachs 78]

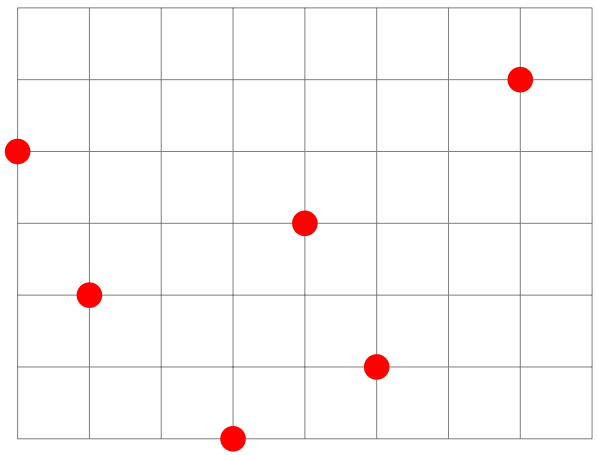
$$x + 2y - 3z + 4 \leq 0 \quad \vee \quad 2x + 7y + 2z \geq 0$$

- **Semialgebraic sets** [Bagnara et al. 05]

$$x^2 + y^2 + z^2 \leq 0 \quad \vee \quad x^2yz^5 - 3yz + 6 \geq 0$$

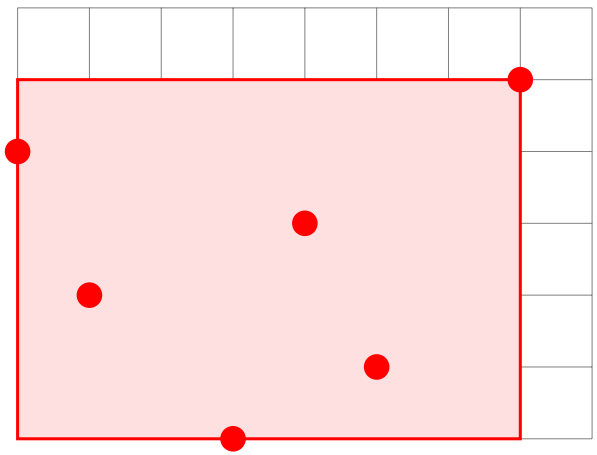
Comparing Abstractions

Original set:



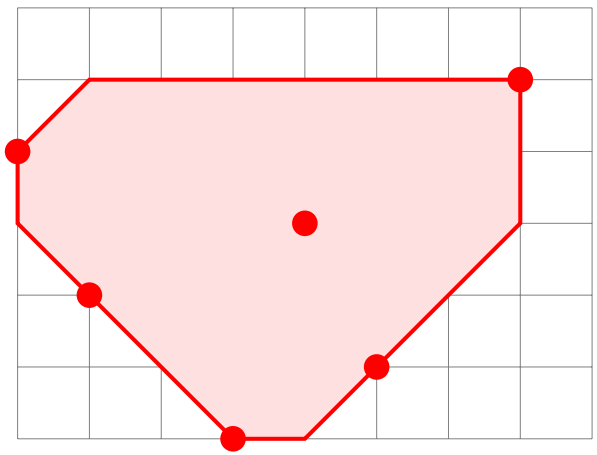
Comparing Abstractions

Interval abstraction:



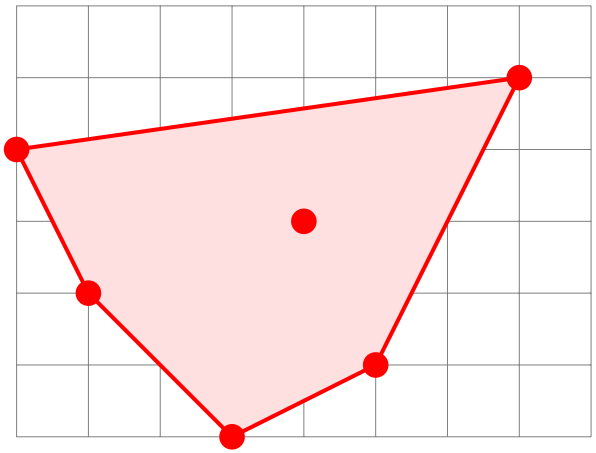
Comparing Abstractions

Octagonal abstraction:



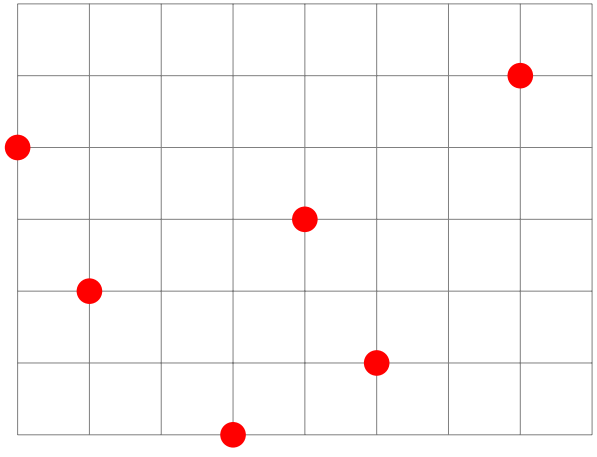
Comparing Abstractions

Polyhedral abstraction:



Comparing Abstractions

Algebraic/semialgebraic/semilinear abstraction :



Comparing Abstractions

$$\begin{array}{ccccccc} \text{Interval} & \leq & \text{Octagonal} & \leq & \text{Semilinear} & \leq & \text{Semialgebraic} \\ & & & & \vee\vert & & \vee\vert \\ & & & & \text{Linear} & \leq & \text{Algebraic} \end{array}$$

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

The loop invariant is:

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

Or equivalently:

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

Does This Program Halt?

$x := 3;$

$y := 2;$

while $2y - x \geq -2$ do

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

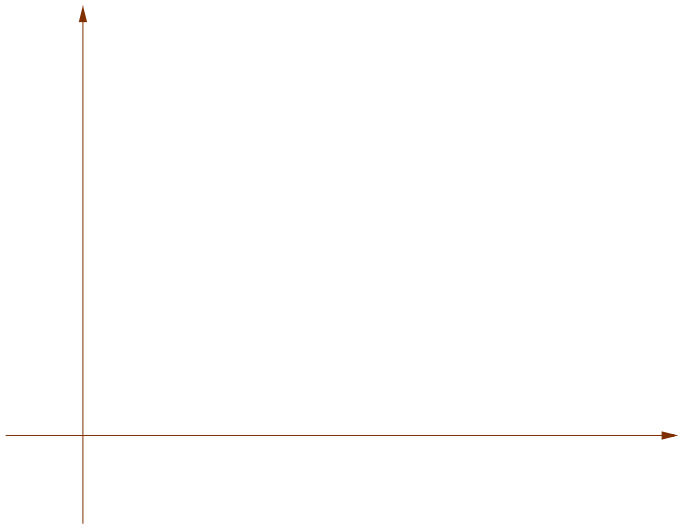
Does This Program Halt?

$x := 3;$

$y := 2;$

while $2y - x \geq -2$ do

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



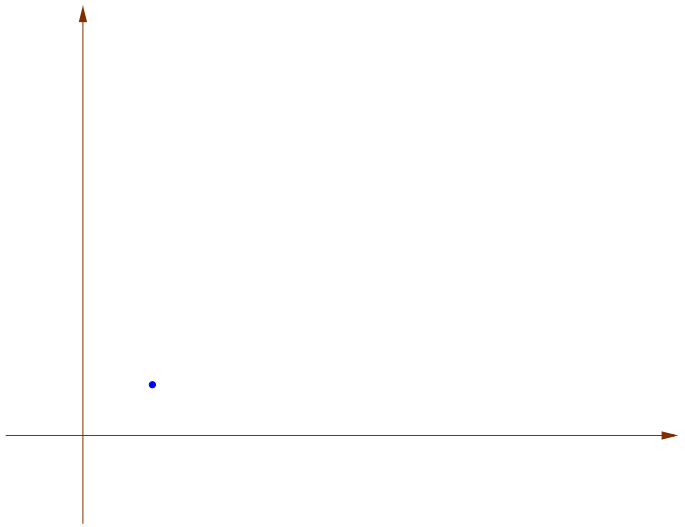
Does This Program Halt?

$x := 3;$

$y := 2;$

while $2y - x \geq -2$ do

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



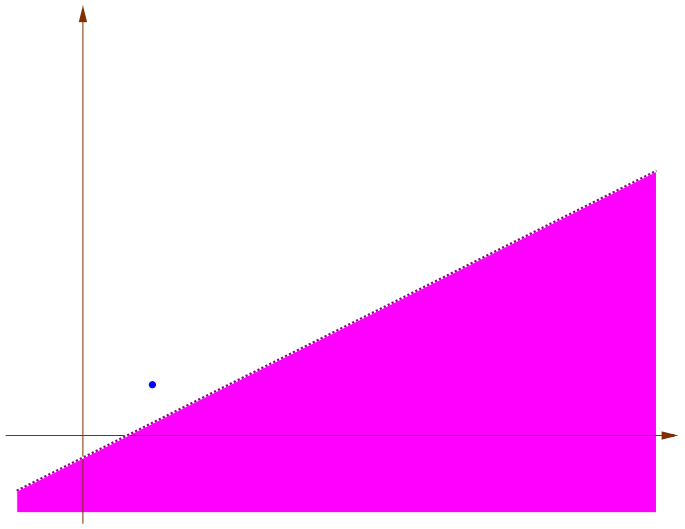
Does This Program Halt?

$x := 3;$

$y := 2;$

while $2y - x \geq -2$ do

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



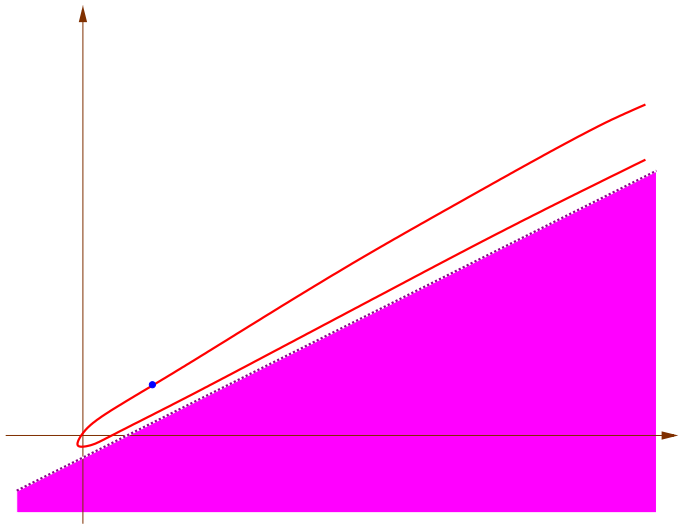
Does This Program Halt?

$x := 3;$

$y := 2;$

while $2y - x \geq -2$ do

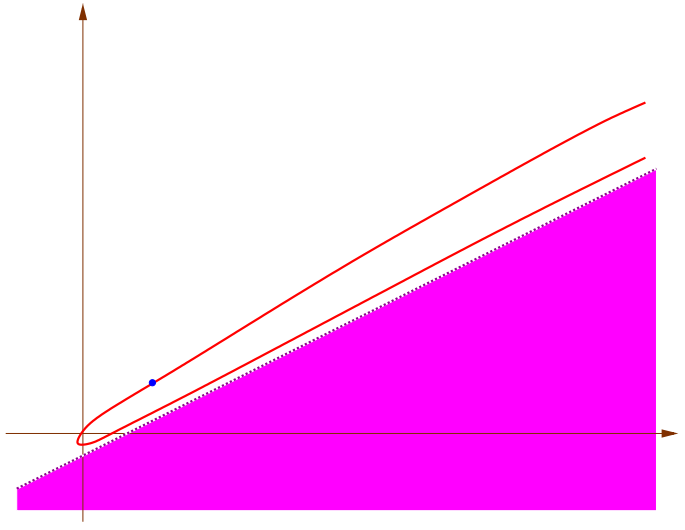
$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 10 & -8 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$



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

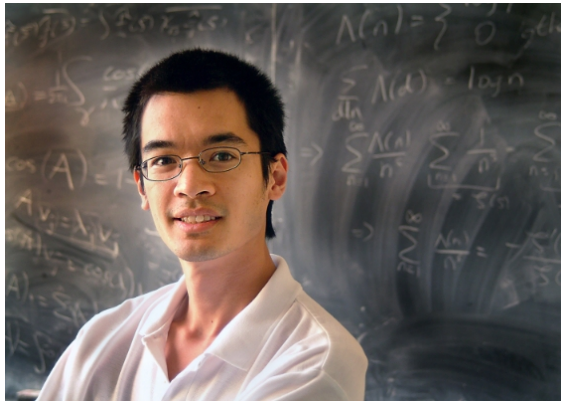
Does This Program Halt?

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

Deciding termination of simple linear loops is open!

"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



A Class of Decision Problems

The Monniaux Problem

Given a program, a safety specification and an abstract domain does there exist an adequate inductive invariant?

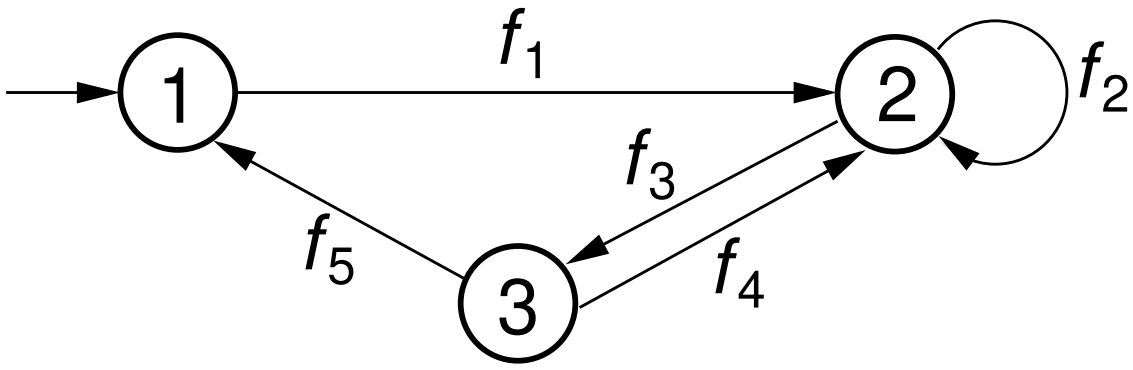
A Class of Decision Problems

The Monniaux Problem

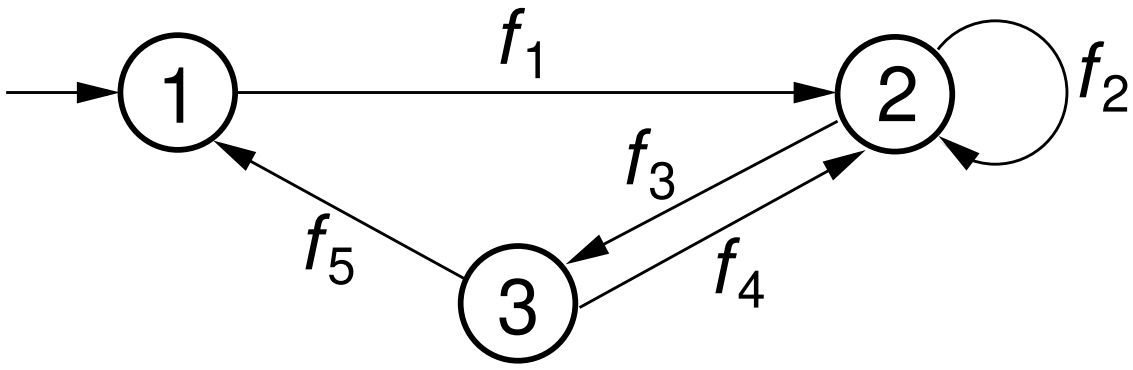
Given a program, a safety specification and an abstract domain does there exist an adequate inductive invariant?

"We started this work hoping to vindicate forty years of research on heuristics by showing that the existence of polyhedral inductive separating invariants in a system with transitions in linear arithmetic (integer or rational) is undecidable."
David Monniaux

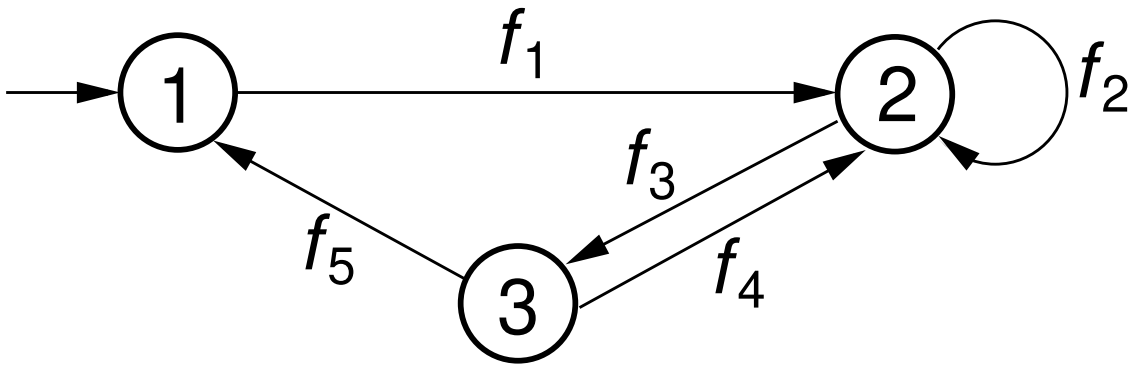
What Are Affine Programs?



What Are Affine Programs?

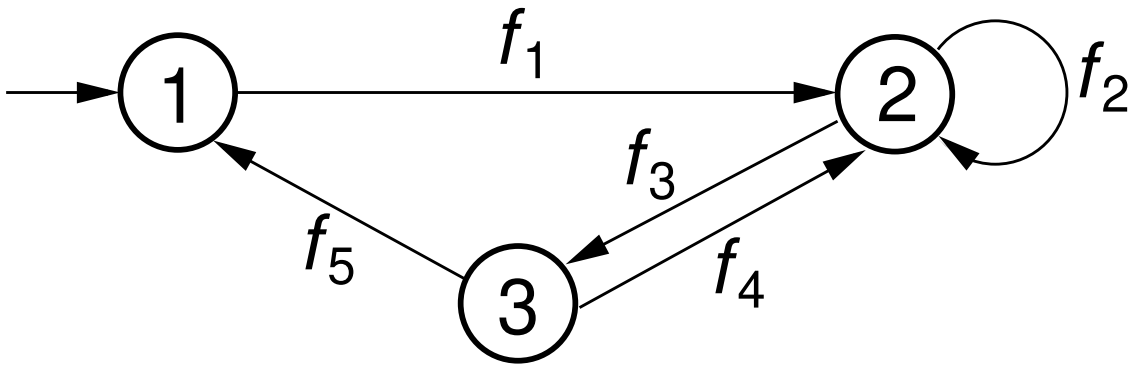


What Are Affine Programs?



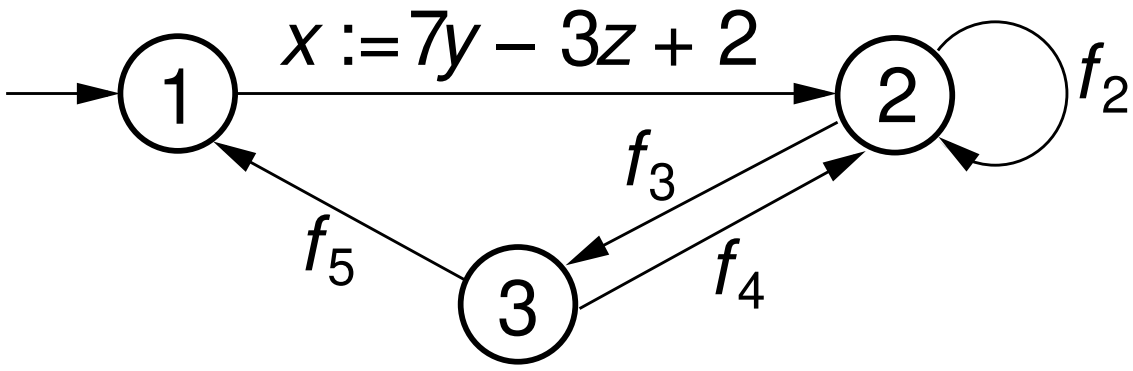
- Only ‘nondeterministic’ branching (no conditionals)

What Are Affine Programs?



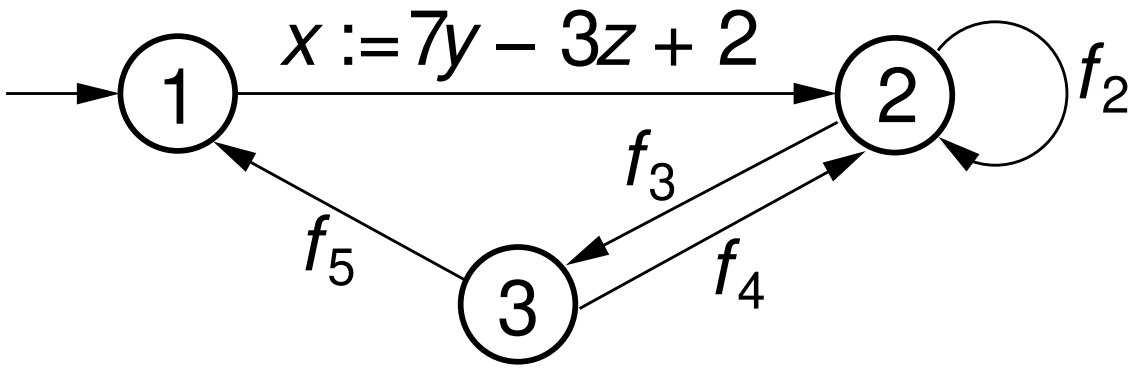
- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine (or linear)

What Are Affine Programs?



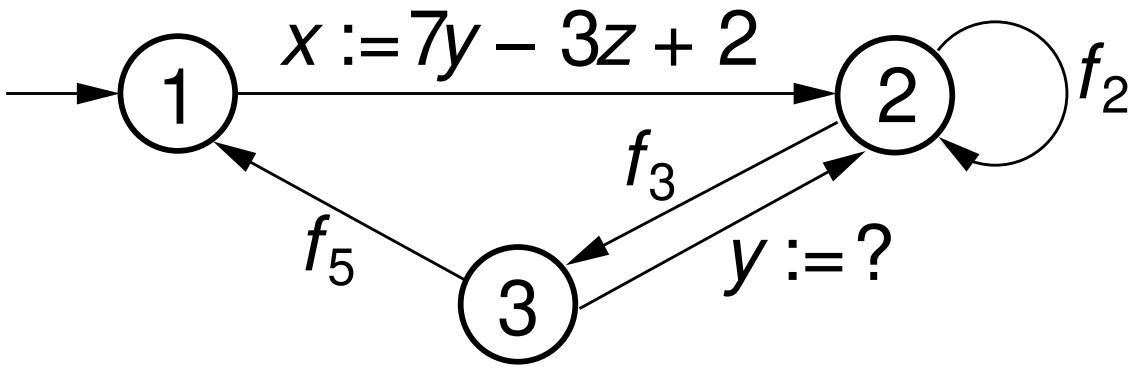
- Only ‘nondeterministic’ branching (no conditionals)
- All assignments are affine (or linear)

What Are Affine Programs?



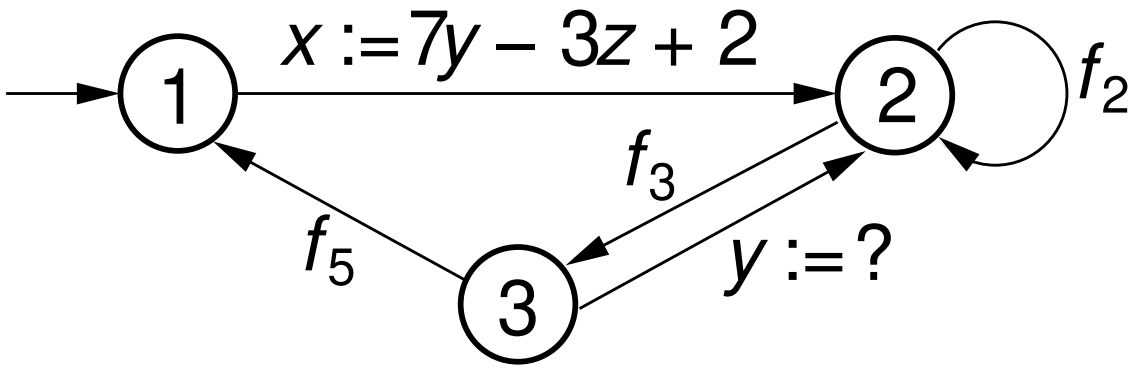
- Only 'nondeterministic' branching (no conditionals)
- All assignments are affine (or linear)
- Also allow nondeterministic assignments $x := ?$

What Are Affine Programs?



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

What Are Affine Programs?

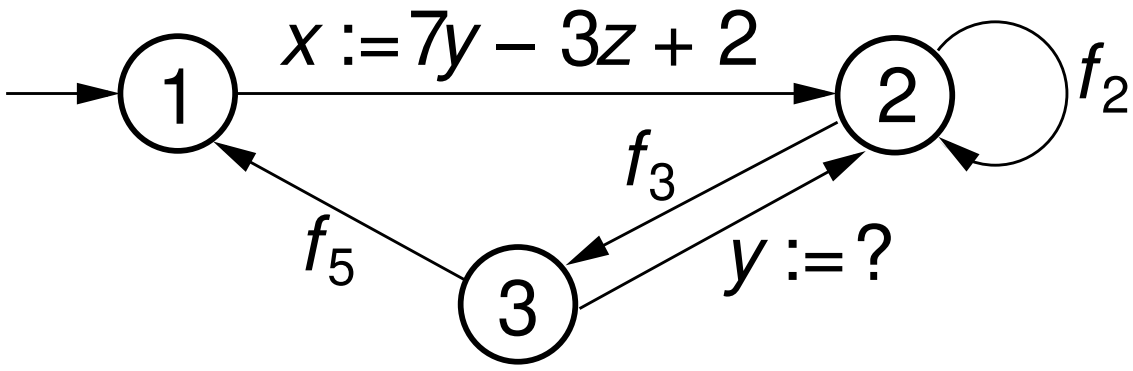


- Only 'nondeterministic' branching (no conditionals)
- All assignments are affine (or linear)
- Also allow nondeterministic assignments $x := ?$

Affine programs:

- can **overapproximate** more complex programs

What Are Affine Programs?

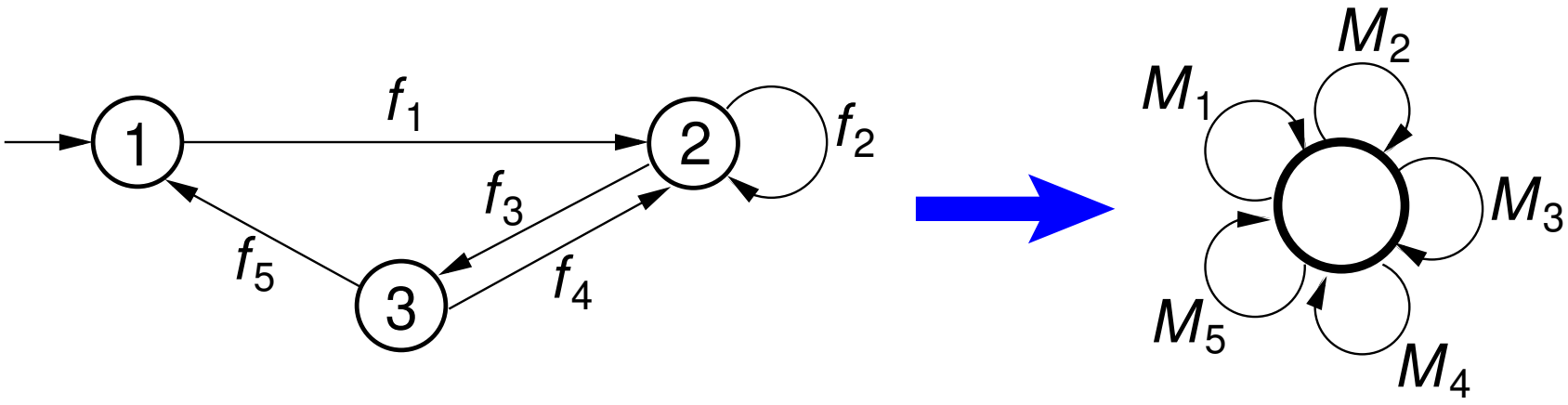


- Only 'nondeterministic' branching (no conditionals)
- All assignments are affine (or linear)
- Also allow nondeterministic assignments $x := ?$

Affine programs:

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

From Affine Programs to Linear Semigroups



each $M_i \in \mathbb{Q}^{d^2}$

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)

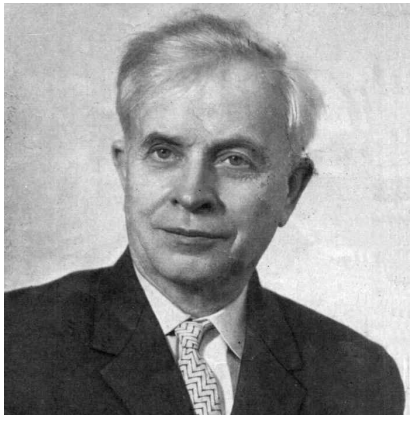
There is a fixed set of 6×6 *integer matrices* $M_1, \ldots, M_k$ *such that the membership problem "*$M \in \langle M_1, \ldots, M_k \rangle$*?" is* **undecidable**.

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.*

State of the Menagerie

- **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

- **Polyhedral / Semilinear sets** [Cousot, Halbwachs 78]

$$x + 2y - 3z + 4 \leq 0 \quad \vee \quad 2x + 7y + 2z \geq 0$$

- **Semialgebraic sets** [Bagnara et al. 05]

$$x^2 + y^2 + z^2 \leq 0 \quad \vee \quad x^2yz^5 - 3yz + 6 \geq 0$$

State of the Menagerie

- ✓ **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- ✓ **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

- **Polyhedral / Semilinear sets** [Cousot, Halbwachs 78]

$$x + 2y - 3z + 4 \leq 0 \quad \vee \quad 2x + 7y + 2z \geq 0$$

- **Semialgebraic sets** [Bagnara et al. 05]

$$x^2 + y^2 + z^2 \leq 0 \quad \vee \quad x^2yz^5 - 3yz + 6 \geq 0$$

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 linear inductive invariant.

Smallest Algebraic Invariants

Theorem (Hrushovski, Ouaknine, Pouly, Worrell 18)

The problem of the existence of algebraic inductive safety invariants for affine program is decidable.

Smallest Algebraic Invariants

Theorem (Hrushovski, Ouaknine, Pouly, Worrell 18)

The problem of the existence of algebraic inductive safety invariants for affine program is decidable.

Smallest algebraic set containing all reachable configurations

$$\Longleftrightarrow$$

Zariski closure of the set of reachable configurations

Smallest Algebraic Invariants

Theorem (Hrushovski, Ouaknine, Pouly, Worrell 18)

The problem of the existence of algebraic inductive safety invariants for affine program is decidable.

Smallest algebraic set containing all reachable configurations

$$\Longleftrightarrow$$

Zariski closure of the set of reachable configurations

Theorem (Hrushovski, Ouaknine, Pouly, Worrell 18)

The presence of guards or polynomial transitions makes the problem undecidable.

State of the Menagerie

- ✓ **Intervals** [Cousot, Cousot 76], [Harrison 77]

$$x \in [0, 4] \quad \wedge \quad y \in [2, \infty)$$

- ✓ **Octagons** [Miné 06]

$$x + y - 2 \leq 2 \quad \wedge \quad x \leq 3 \quad \wedge \quad y - x \leq 1$$

- ✓ **Linear / Algebraic sets** [Müller-Olm, Seidl 04]

$$x^3 - y^2 = 0 \quad \wedge \quad x^2yz^5 - 3yz = 0$$

- **Polyhedral / Semilinear sets** [Cousot, Halbwachs 78]

$$x + 2y - 3z + 4 \leq 0 \quad \vee \quad 2x + 7y + 2z \geq 0$$

- **Semialgebraic sets** [Bagnara et al. 05]

$$x^2 + y^2 + z^2 \leq 0 \quad \vee \quad x^2yz^5 - 3yz + 6 \geq 0$$

Undecidability for Semilinear invariants

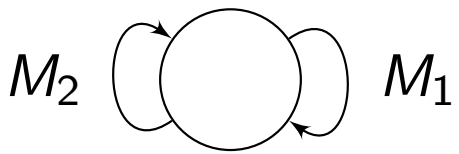
Theorem

The problem of the existence of semilinear safety invariants is undecidable.

Undecidability for Semilinear invariants

Theorem

The problem of the existence of semilinear safety invariants is undecidable.

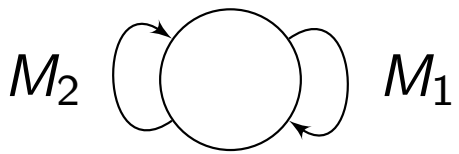


- Only two transitions required (matrices of size 336).
- A single "bad" point.

Undecidability for Semilinear invariants

Theorem

The problem of the existence of semilinear safety invariants is undecidable.

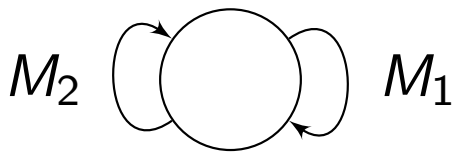


- Only two transitions required (matrices of size 336).
- A single "bad" point.
- Reachability of the "bad" point is not possible.

Undecidability for Semilinear invariants

Theorem

The problem of the existence of semilinear safety invariants is undecidable.



- Only two transitions required (matrices of size 336).
- A single "bad" point.
- Reachability of the "bad" point is not possible.

Invariant of the form: $I = S \cup F$
S is a simple safe set.
F is a finite number of points.

Decidability for Single Loop

While $x \notin Bad$ **do**
$x := Mx$

Decidability for Single Loop

While $x \notin Bad$ **do**
$x := Mx$

Theorem

The problem of the existence of semilinear safety invariants for while loop is decidable.

Decidability for Single Loop

While $x \notin Bad$ **do**
$x := Mx$

Theorem

The problem of the existence of semilinear safety invariants for while loop is decidable.

- Build the Jordan normal form;

Decidability for Single Loop

While $x \notin Bad$ **do**
$x := Mx$

Theorem

The problem of the existence of semilinear safety invariants for while loop is decidable.

- Build the Jordan normal form;
- Build invariant if there exist "simple" eigenvalues;

Decidability for Single Loop

While $x \notin Bad$ **do**
$x := Mx$

Theorem

The problem of the existence of semilinear safety invariants for while loop is decidable.

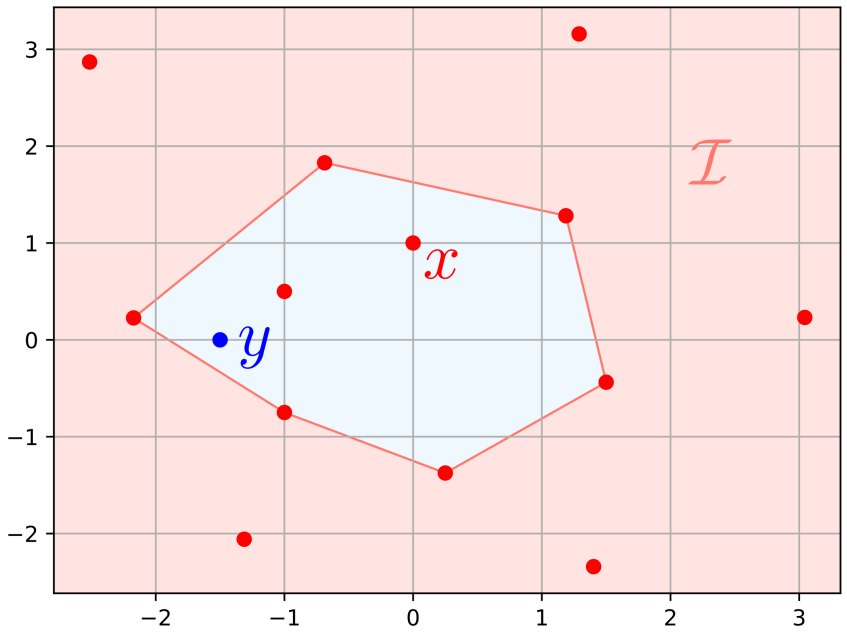
- Build the Jordan normal form;
- Build invariant if there exist "simple" eigenvalues;
- Prove the absence of non-trivial semilinear invariant otherwise.

Case $|\lambda| > 1$

Starting in x

Bad $= y$

Sequence in the eigenspace is a diverging spiral

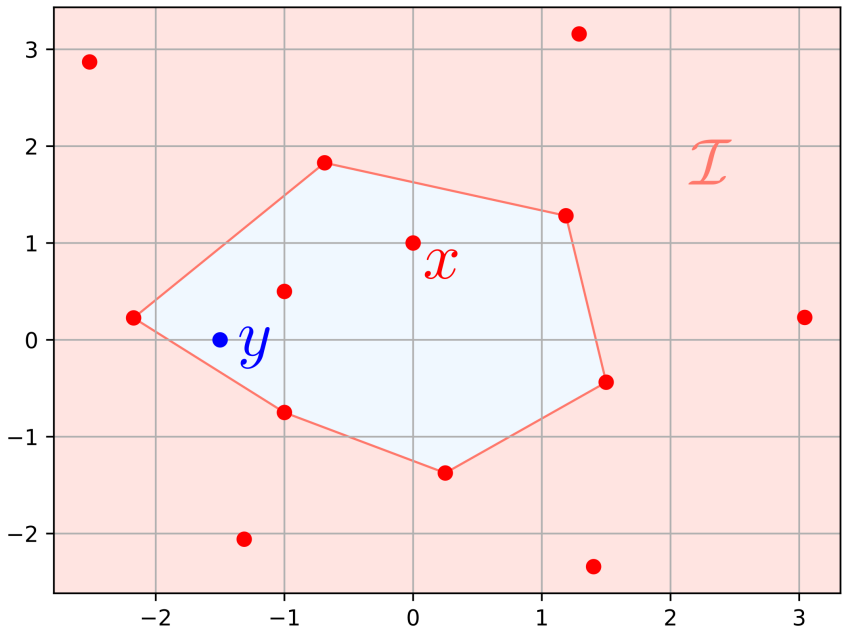


Case $|\lambda| > 1$

Starting in x

Bad $= y$

Sequence in the eigenspace is a diverging spiral



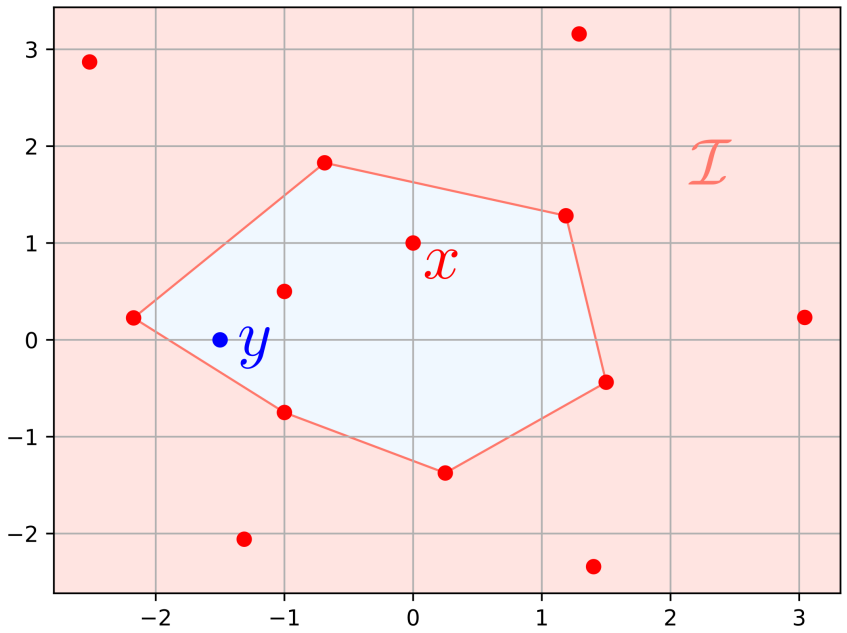
Most difficult case: modulus 1 and not root of unity.

Case $|\lambda| > 1$

Starting in x

Bad $= y$

Sequence in the eigenspace is a diverging spiral



Most difficult case: modulus 1 and not root of unity.

$\rightarrow$ Only semilinear invariant given by relations between eigenvalues

What about Convex Invariant?

- ✓ simplicity of representation and implementation
- ✓ algorithmic tractability and scalability
- ✓ good termination heuristics

What about Convex Invariant?

- ✓ simplicity of representation and implementation
- ✓ algorithmic tractability and scalability
- ✓ good termination heuristics
- ✗ lack of expressivity

What about Convex Invariant?

- ✓ simplicity of representation and implementation
- ✓ algorithmic tractability and scalability
- ✓ good termination heuristics
- ✗ lack of expressivity

Theorem (Monniaux 19)

The problem of the existence of convex semilinear safety invariants for affine programs with polynomial guards is undecidable.

What about Convex Invariant?

- ✓ simplicity of representation and implementation
- ✓ algorithmic tractability and scalability
- ✓ good termination heuristics
- ✗ lack of expressivity

Theorem (Monniaux 19)

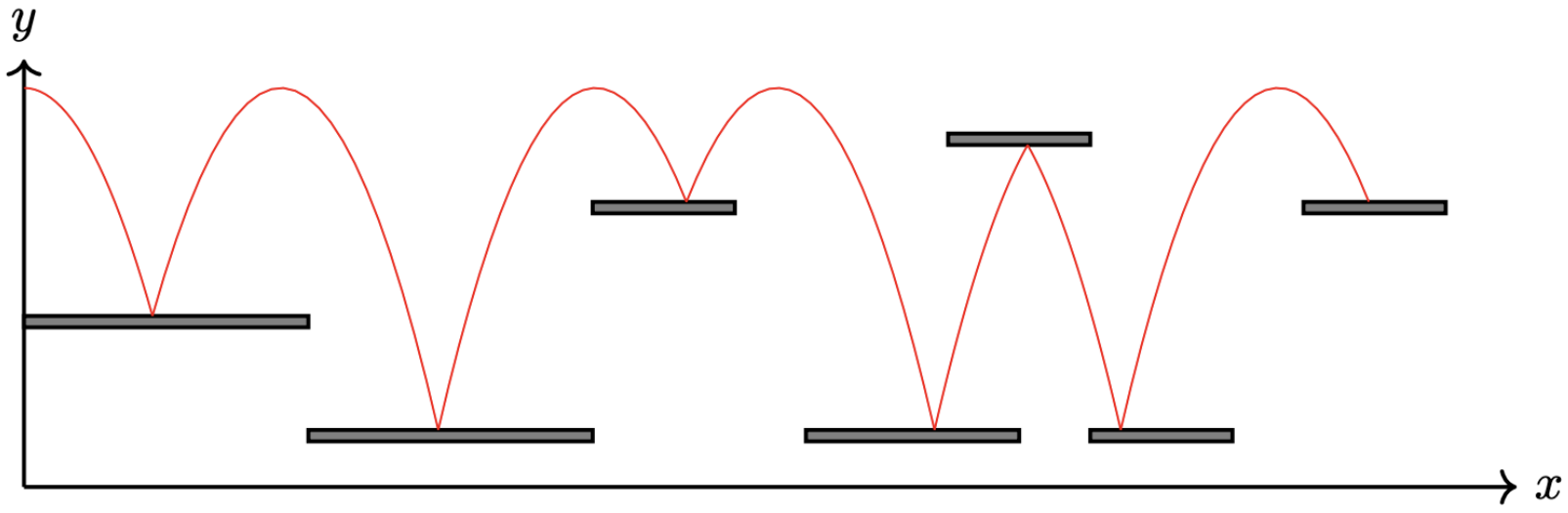
The problem of the existence of convex semilinear safety invariants for affine programs with polynomial guards is undecidable.

The general case remains open and challenging.

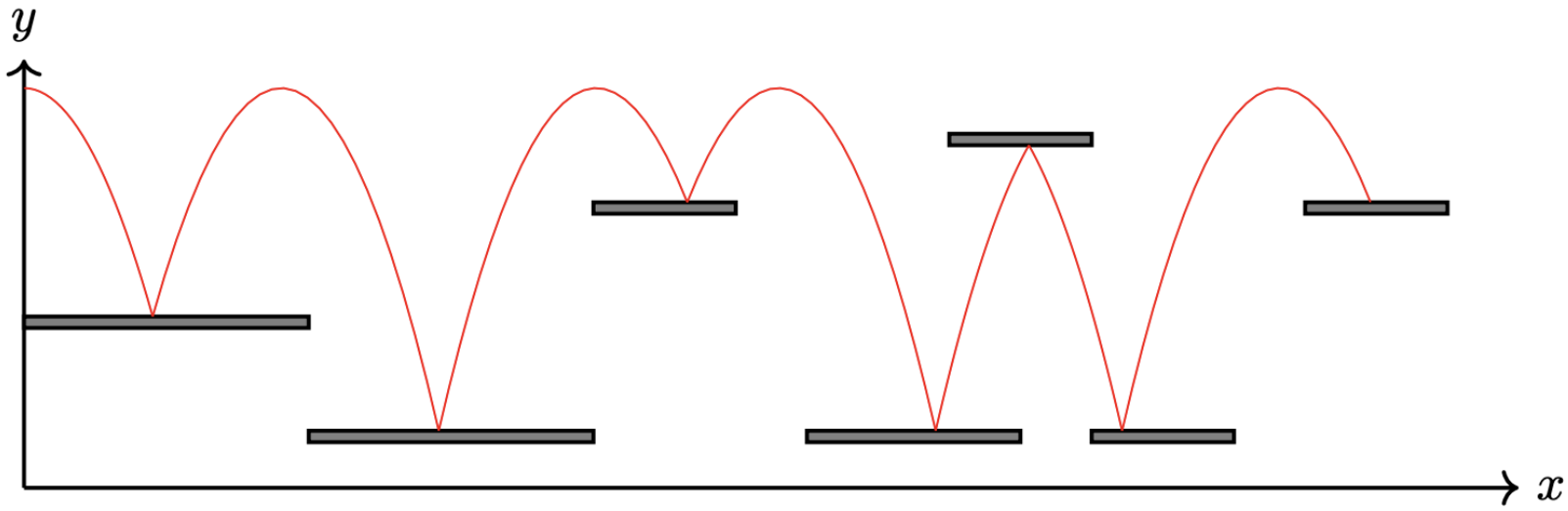
Ongoing Research Programme

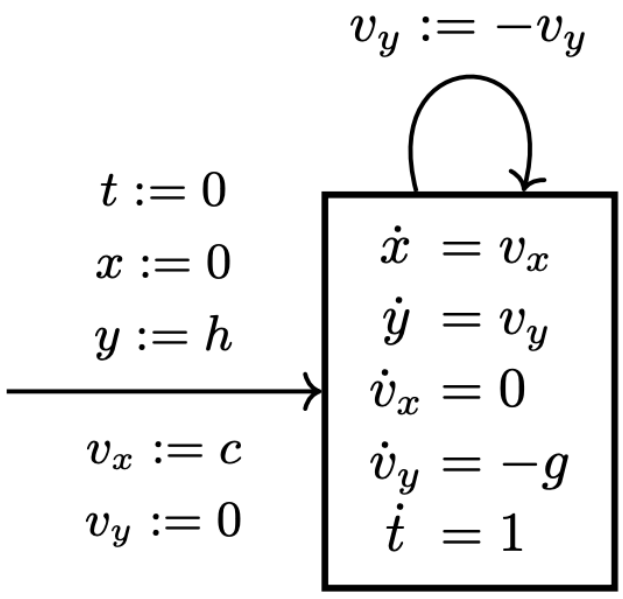
- The Monniaux Problem for convex invariants
- Orbit-finiteness for *polynomial* programs
- Algebraic and semialgebraic invariants for continuous dynamical systems & hybrid automata

A Bouncing Ball

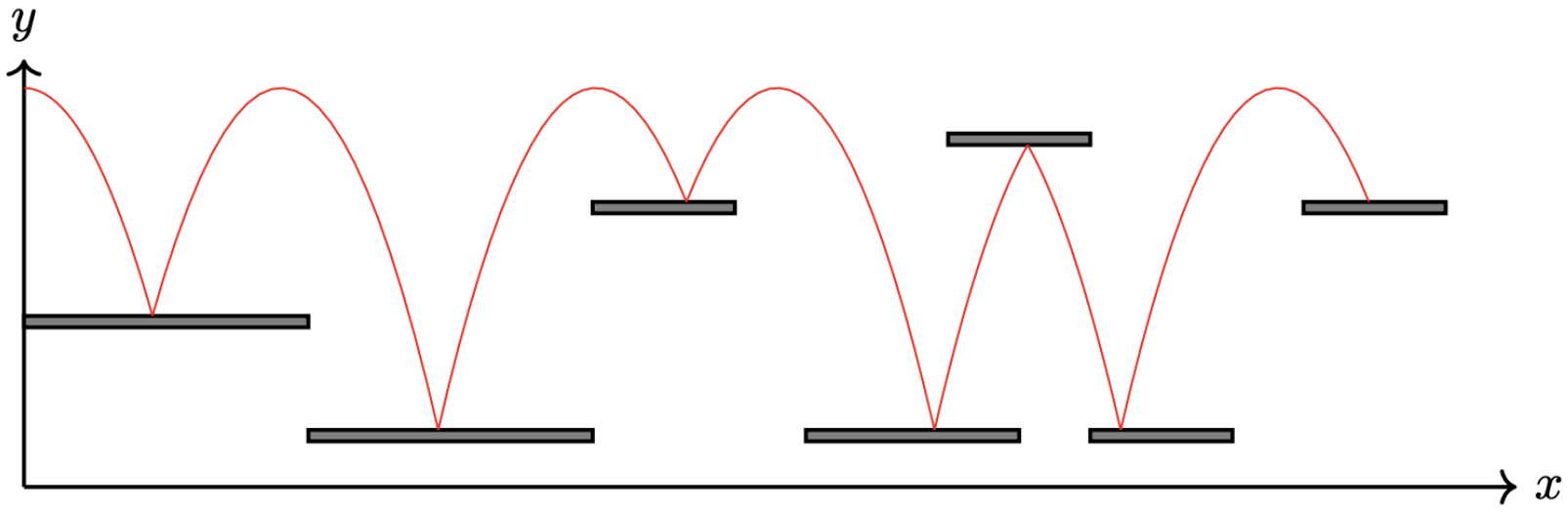


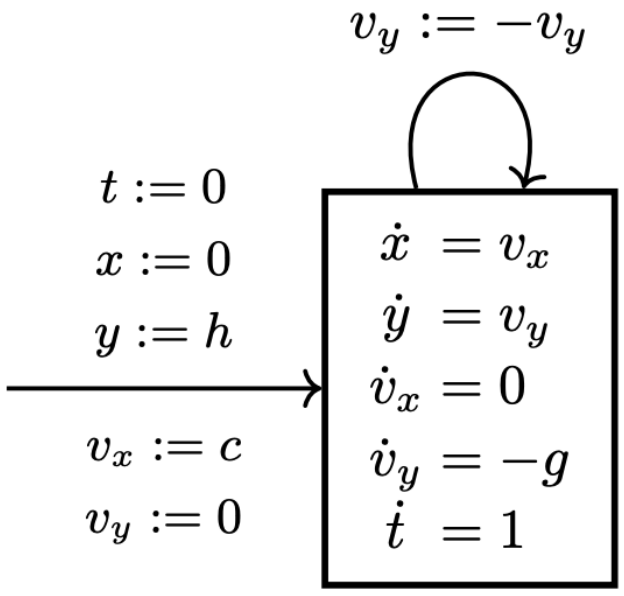
A Linear Hybrid Automaton (LHA)





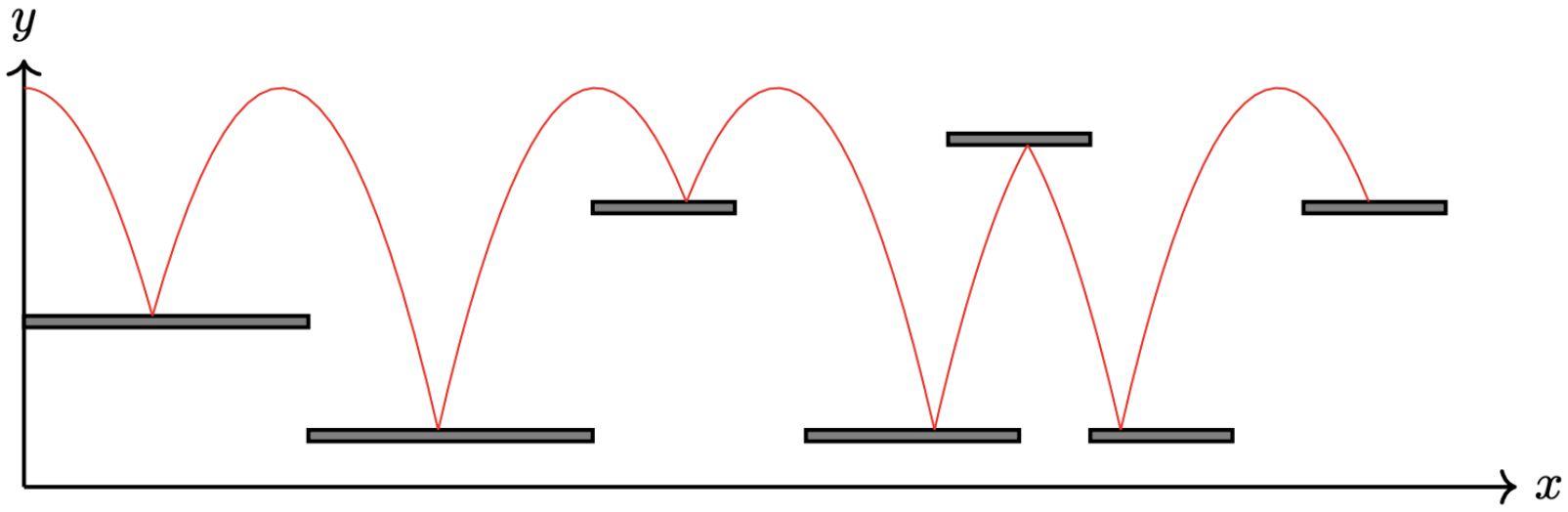
Strongest Algebraic Invariants for LHA

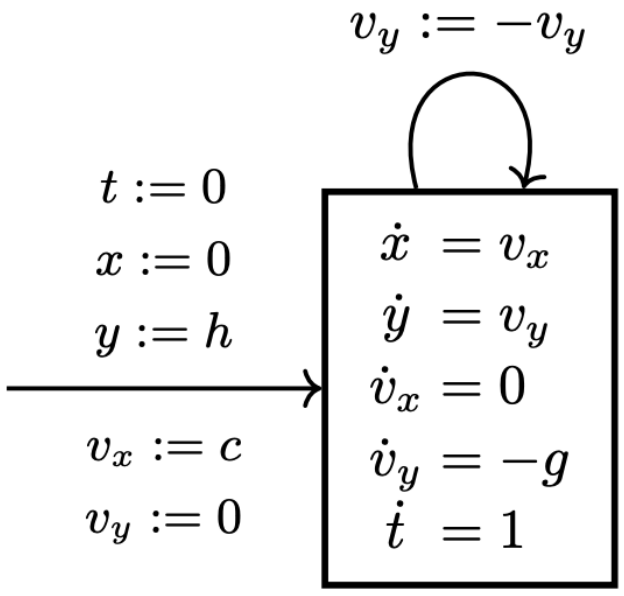




$v_x = c$

Strongest Algebraic Invariants for LHA

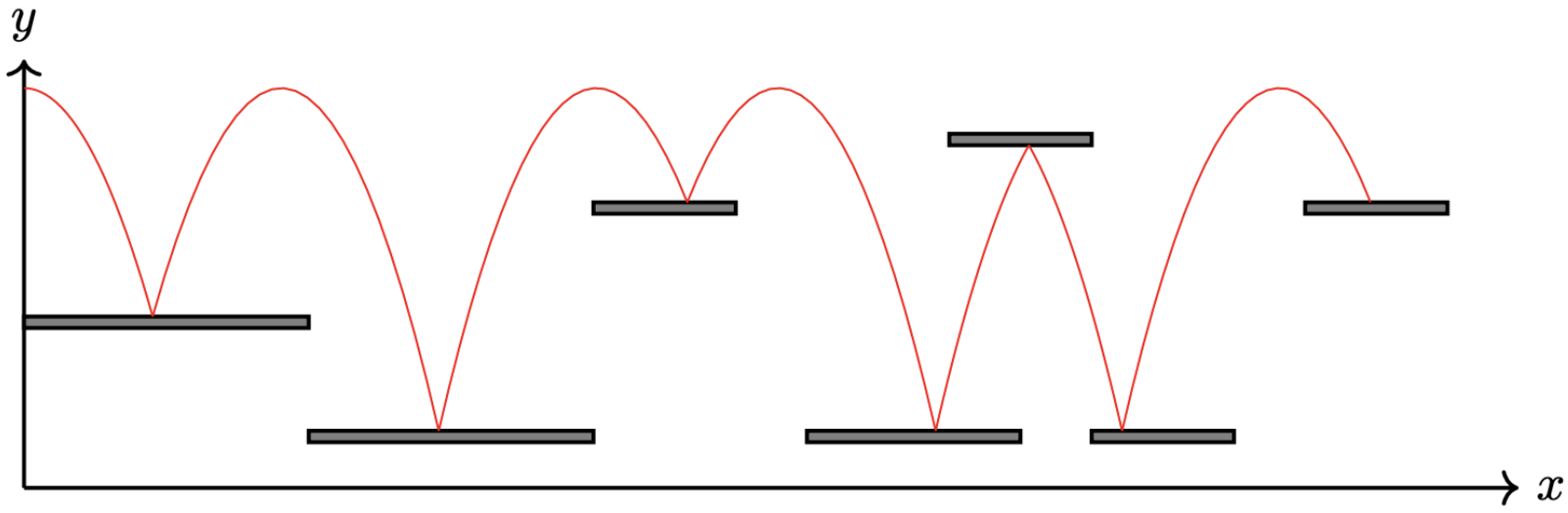


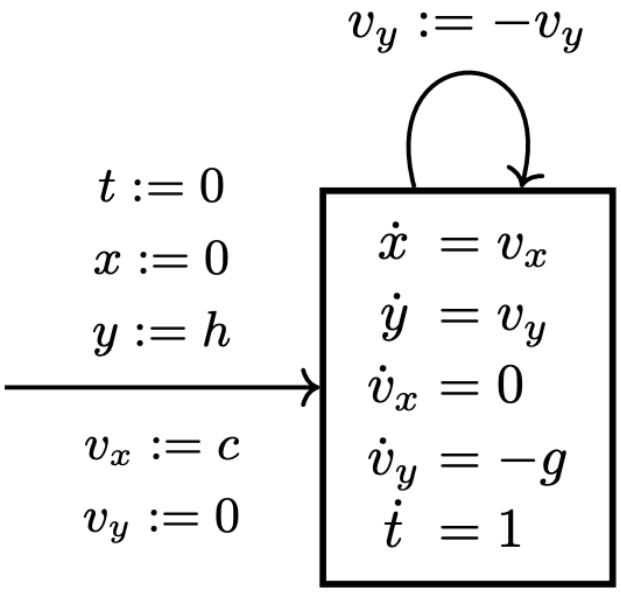


$v_x = c$

$x = tc$

Strongest Algebraic Invariants for LHA





$v_x = c$

$x = tc$

$v_y{}^2 + 2g(y - h) = 0$