Static Analysis of Control Command Systems: Synthesizing non Linear Invariants

Pierre Roux [1,2]

December 18, 2013

[1]ONERA – The French Aerospace Lab, Toulouse, France

[2]ISAE, University of Toulouse, Toulouse, France

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE

ENS DE LYON

ISAE
Institut Supérieur de l'Aéronautique et de l'Espace

ONERA
THE FRENCH AEROSPACE LAB

Digital Flight Commands

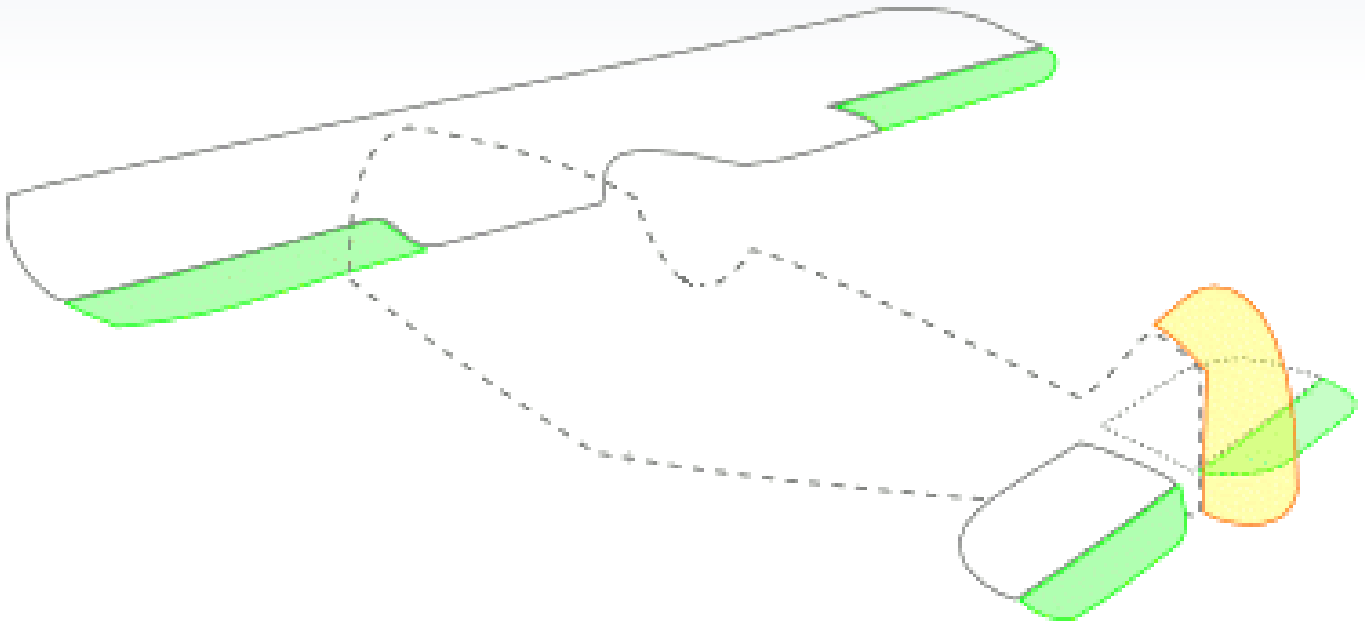


Image: Piotr Jaworski (GFDL)

Digital Flight Commands

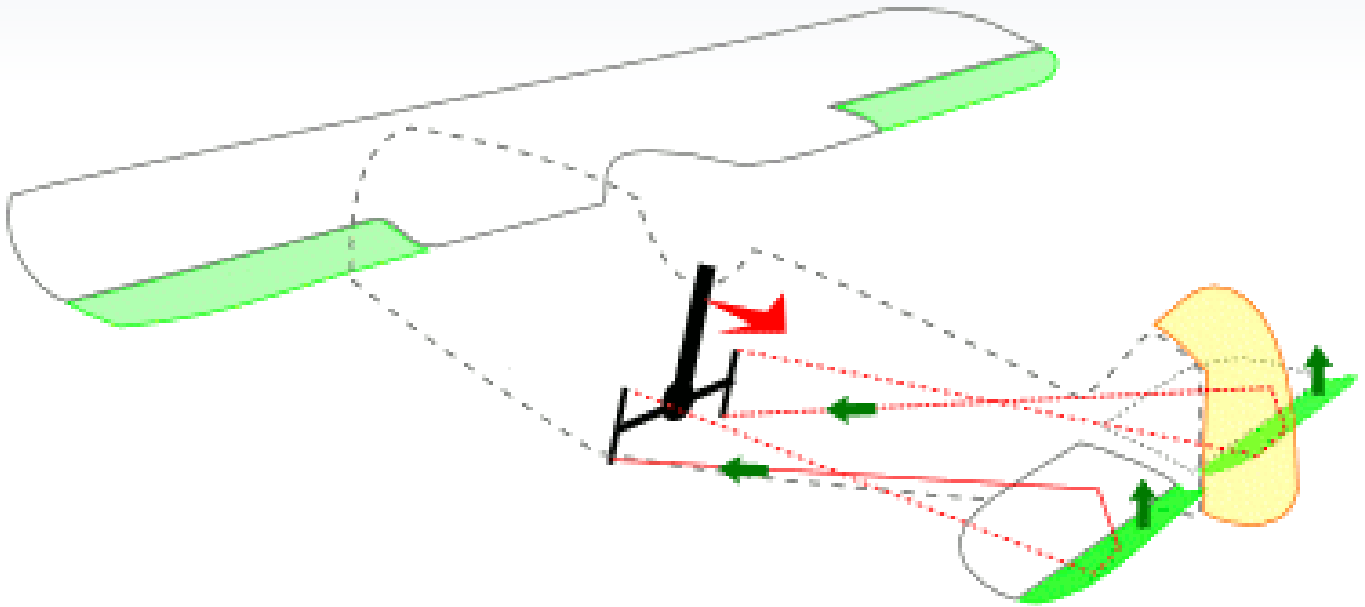


Image: Piotr Jaworski (GFDL)

Cables

Image: public domain

Digital Flight Commands

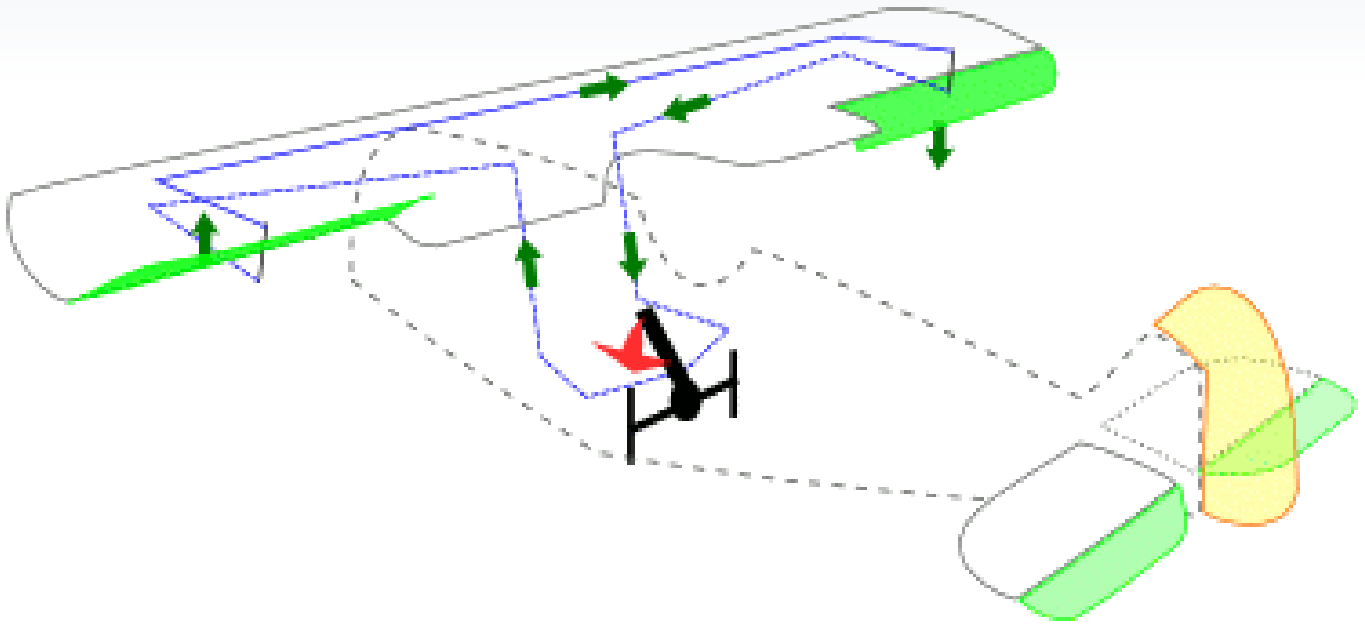


Image: Piotr Jaworski (GFDL)

Cables

Image: public domain

Digital Flight Commands

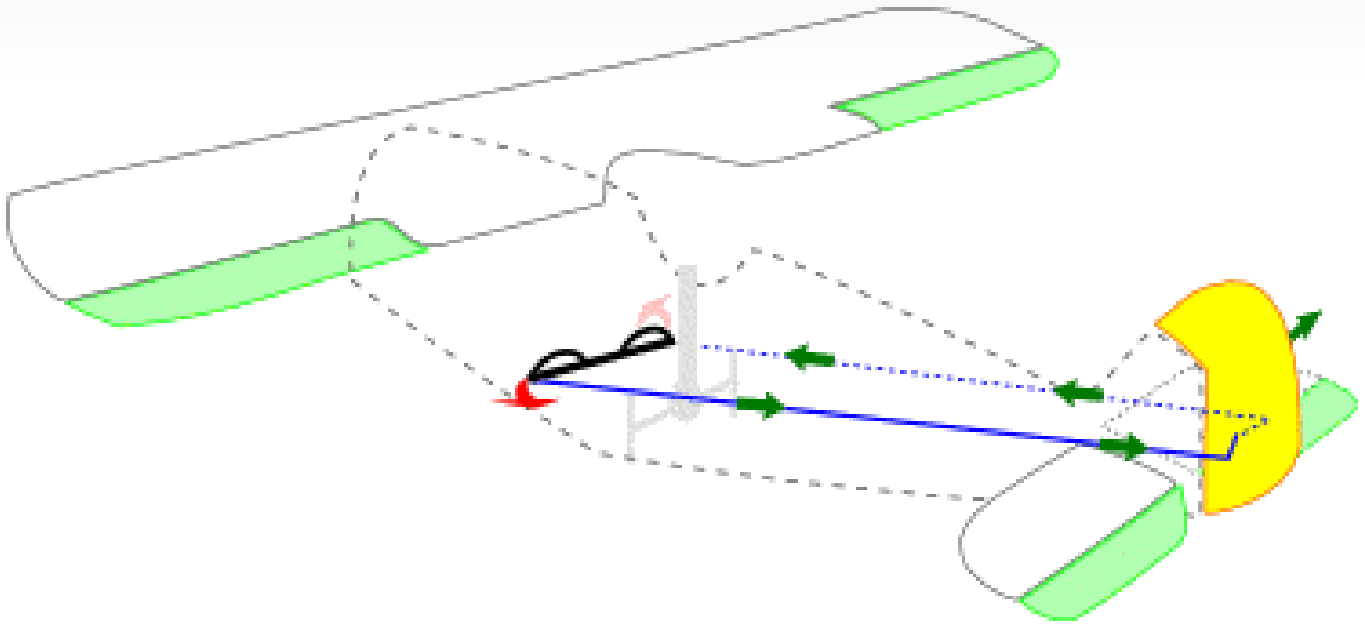


Image: Piotr Jaworski (GFDL)

Cables

Image: public domain

Digital Flight Commands

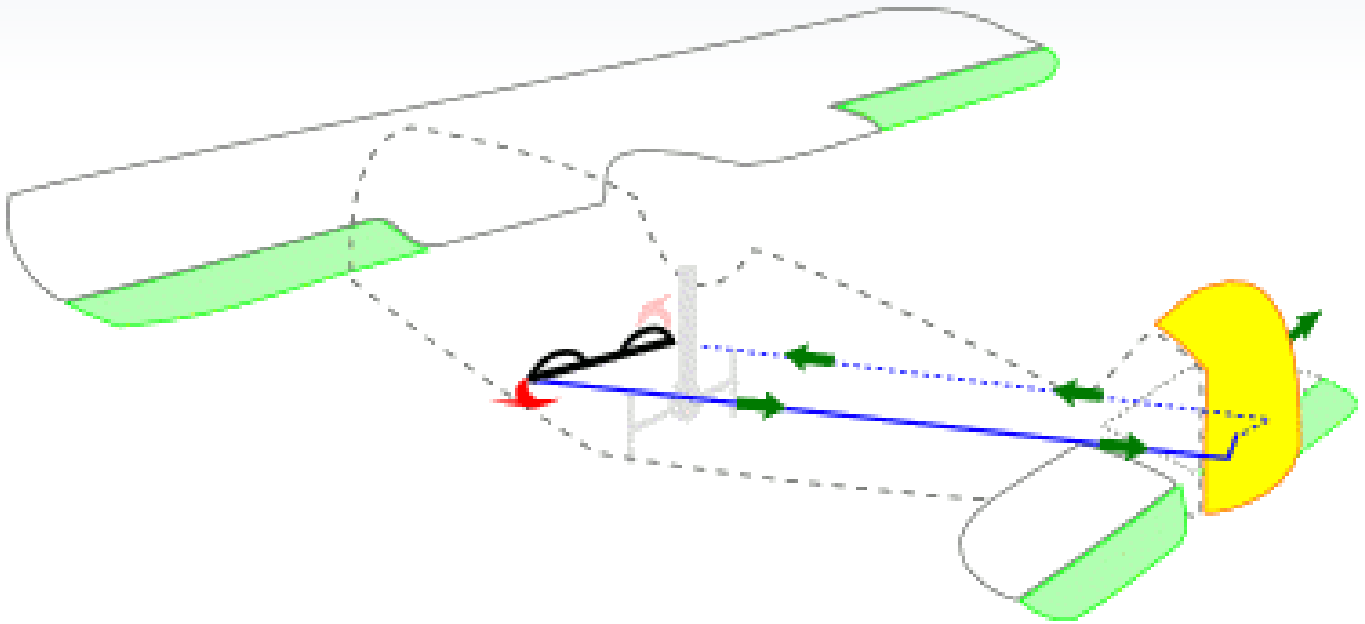


Image: Piotr Jaworski (GFDL)

Cables

Image: public domain

Hydraulic Actuators

Image: Woodward

Digital Flight Commands

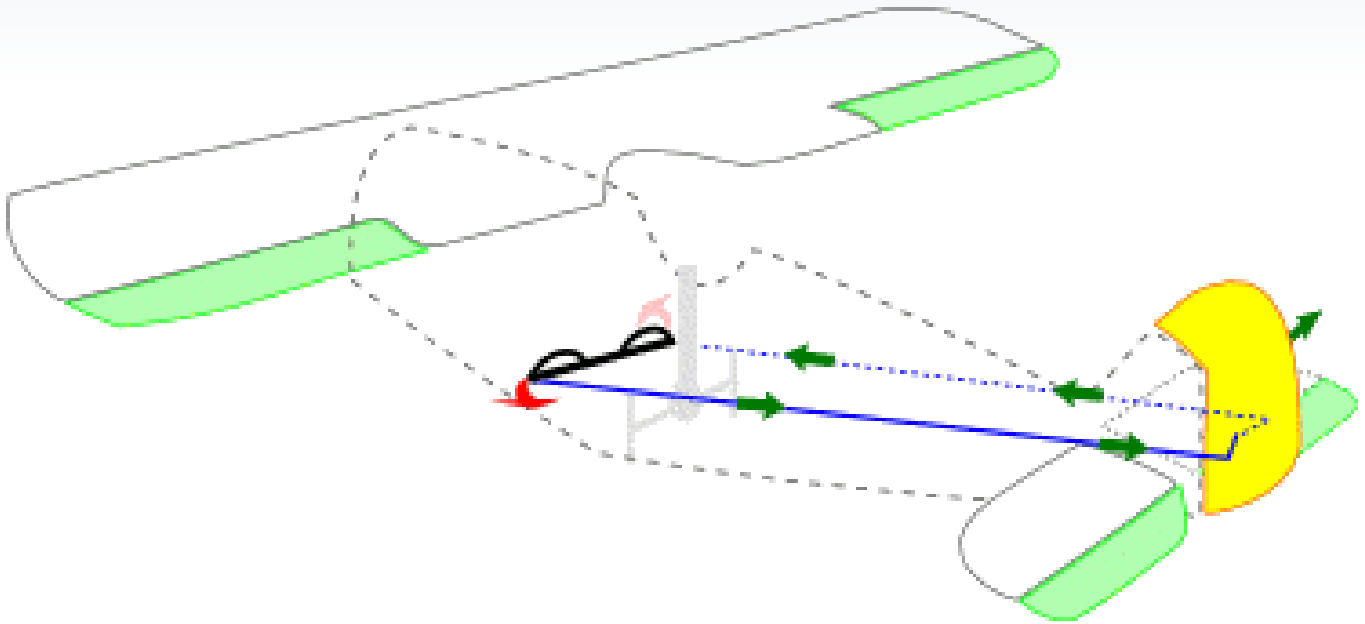


Image: Piotr Jaworski (GFDL)

Cables

Image: public domain

Hydraulic Actuators

Image: Woodward

Computers

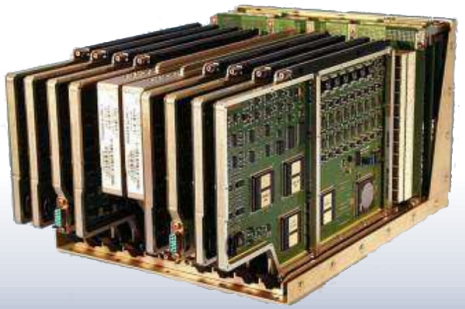


Image: Airbus

Control Command Systems

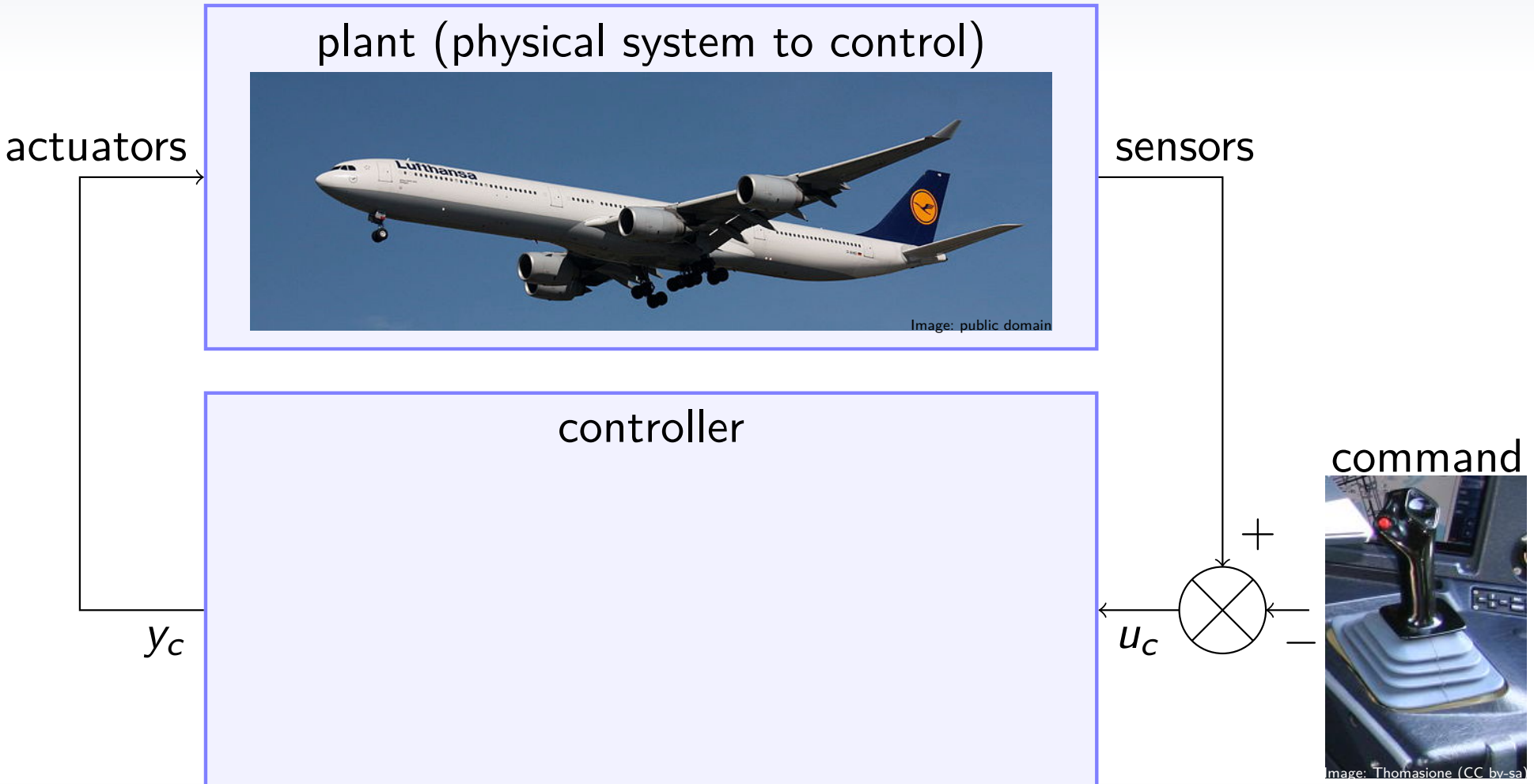
plant (physical system to control)

Image: public domain

command

Image: Thomasione (CC by-sa)

Control Command Systems



Control Command Systems

plant (physical system to control)

Image: public domain

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

u_c

command

Image: Thomasione (CC by-sa)

Control Command Systems

plant (physical system to control)

Image: public domain

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

u_c

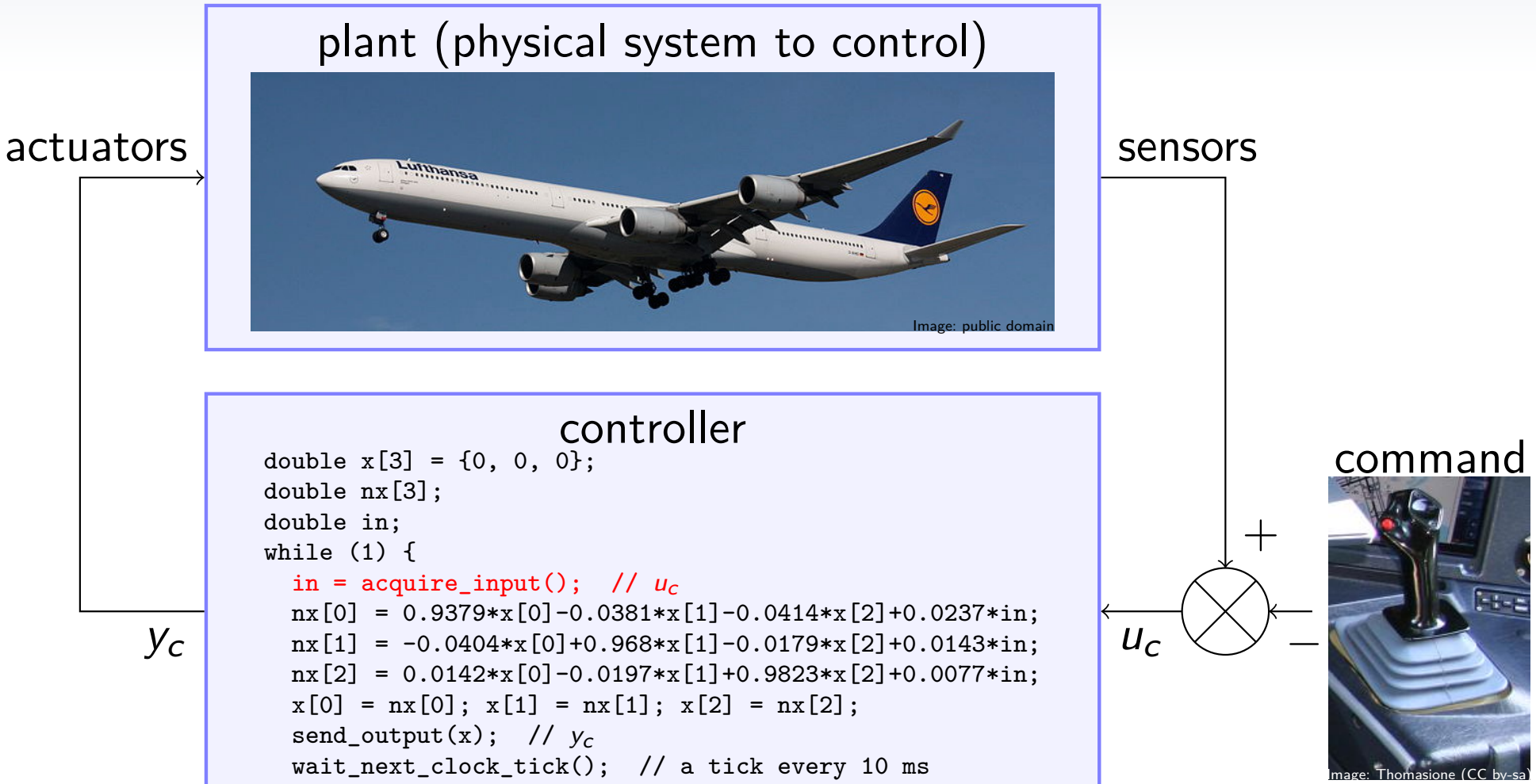
+

−

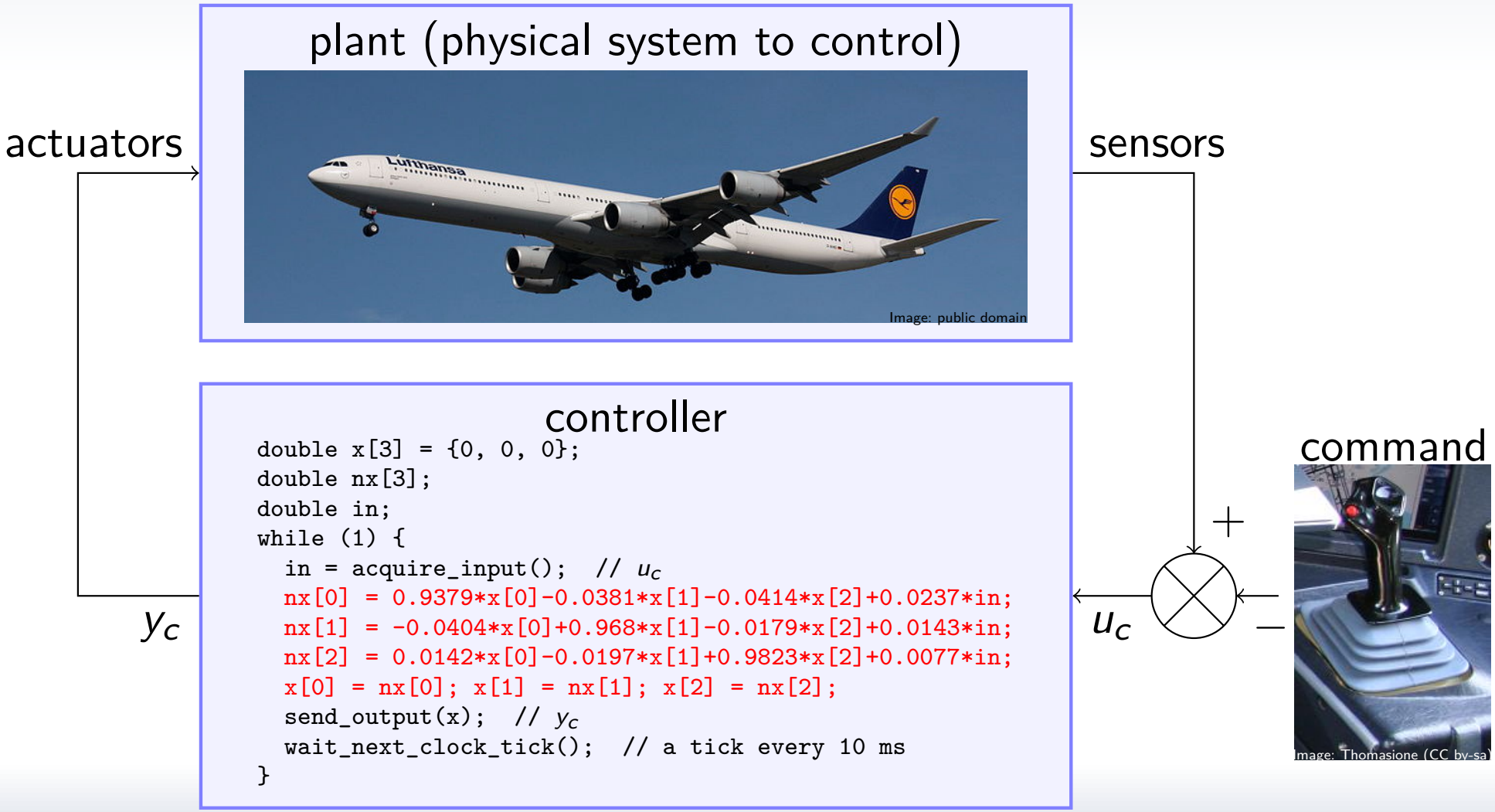
command

Image: Thomasione (CC by-sa)

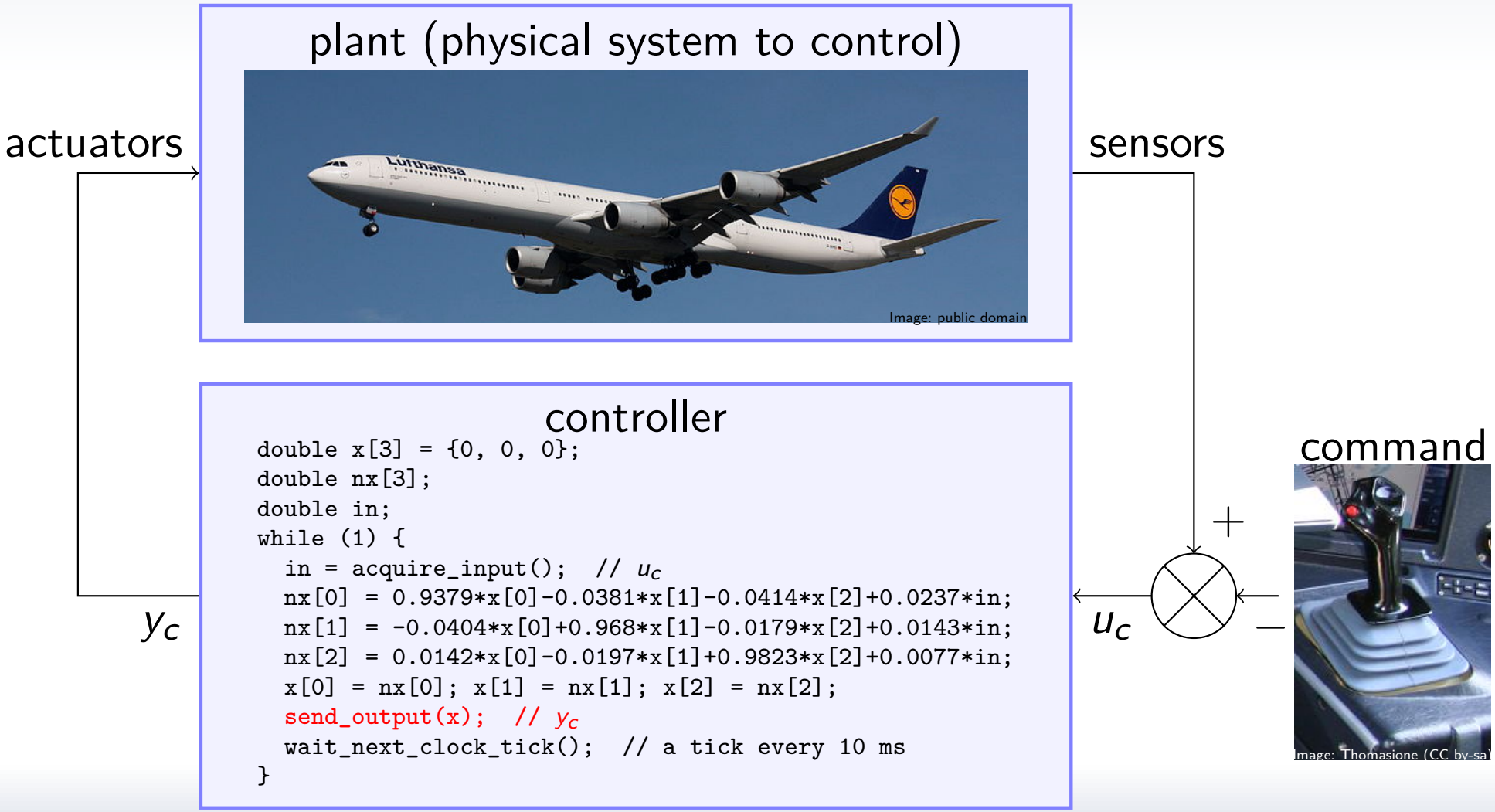
Control Command Systems



Control Command Systems



Control Command Systems



Control Command Systems

plant (physical system to control)

Image: public domain

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

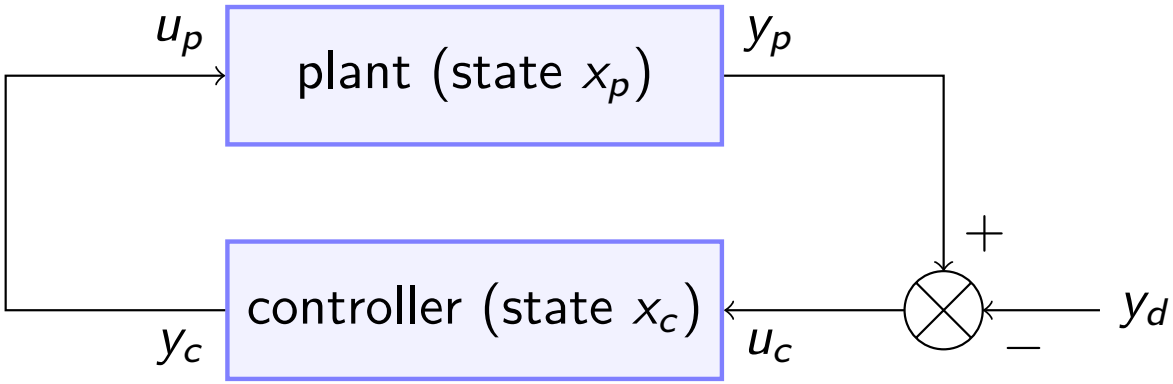
u_c

command

Image: Thomasione (CC by-sa)

Stability Proofs

- Closed loop stability

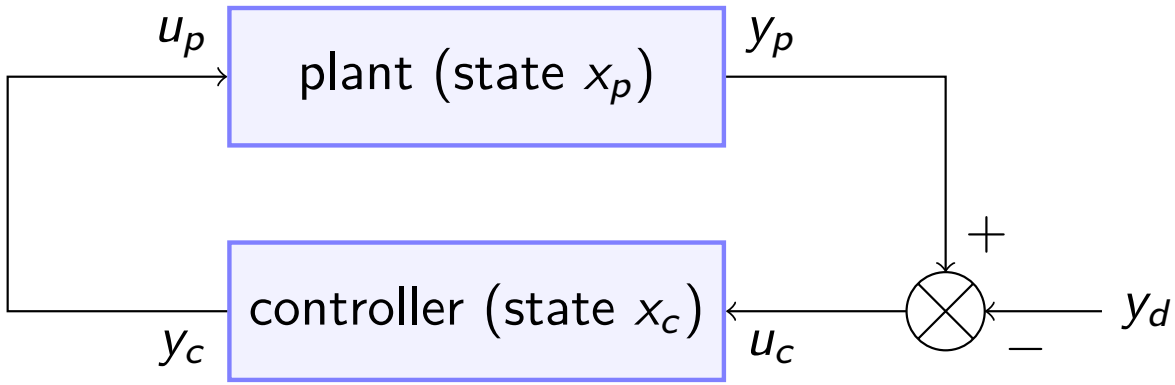


command y_d bounded $\Rightarrow$ x_c and x_p bounded

(hence y_c and y_p bounded)

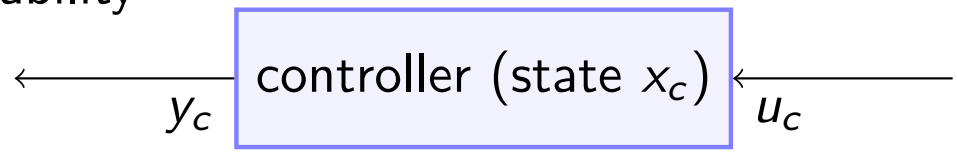
Stability Proofs

- Closed loop stability



command y_d bounded $\Rightarrow$ x_c and x_p bounded
(hence y_c and y_p bounded)

- Open loop stability



input u_c bounded $\Rightarrow$ x_c bounded (hence y_c bounded)

Static Analysis

The controller

x0 := 0; x1 := 0; x2 := 0;
while $-1 \leq 0$ **do**
  in := **?(**-1, 1**)**;
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od

is open loop stable: $|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

Static Analysis

The controller

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

is open loop stable: $|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

Goal

Design a program (**static analyzer**)
to compute these bounds from above source code.

Static Analysis

The controller

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

is open loop stable: $|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

Goal

Design a program (**static analyzer**)
to compute these bounds from above source code.

Static Analysis

The controller

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

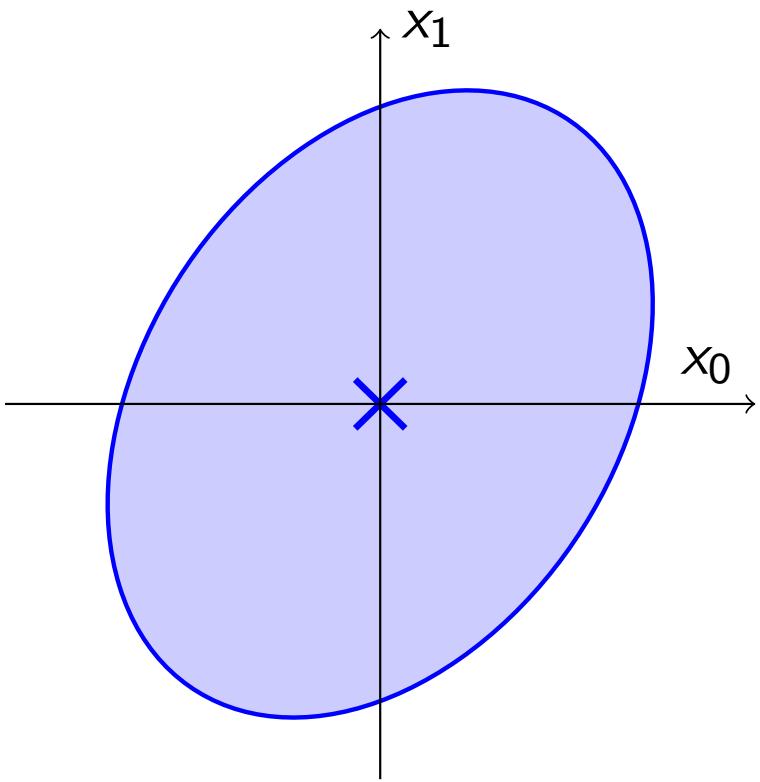
is open loop stable: $|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

Goal

Design a program (**static analyzer**)
to compute these bounds from above source code.

Inductive Invariants

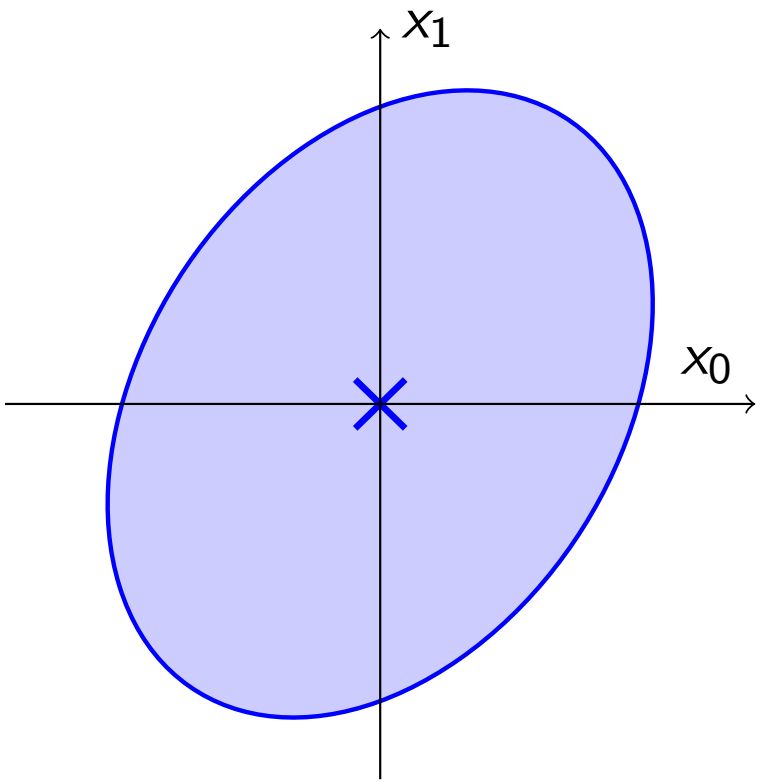
A set of points is an inductive invariant for a controller if it:



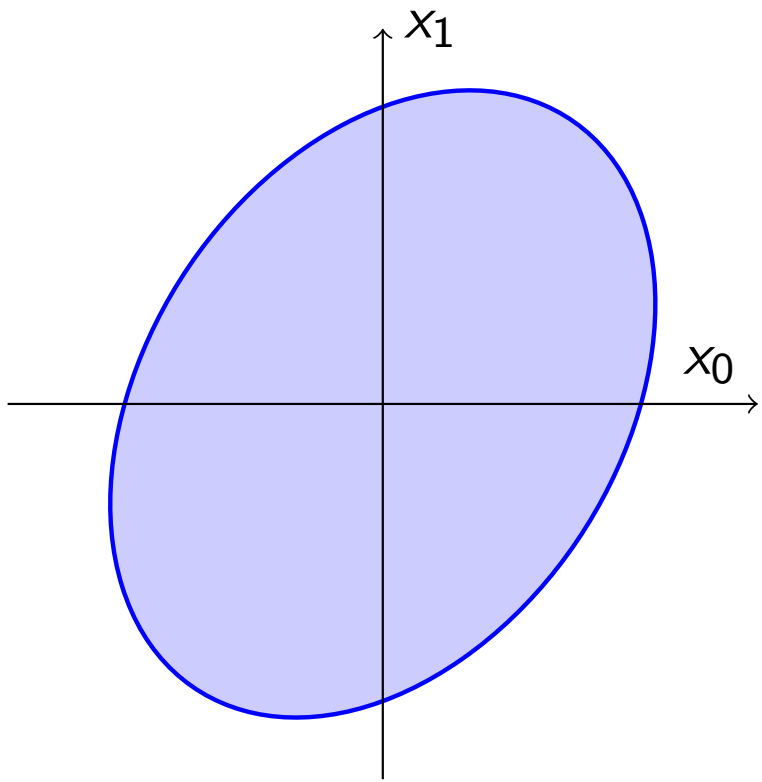
contains initial state
$((0, 0)$ for instance)

Inductive Invariants

A set of points is an inductive invariant for a controller if it:



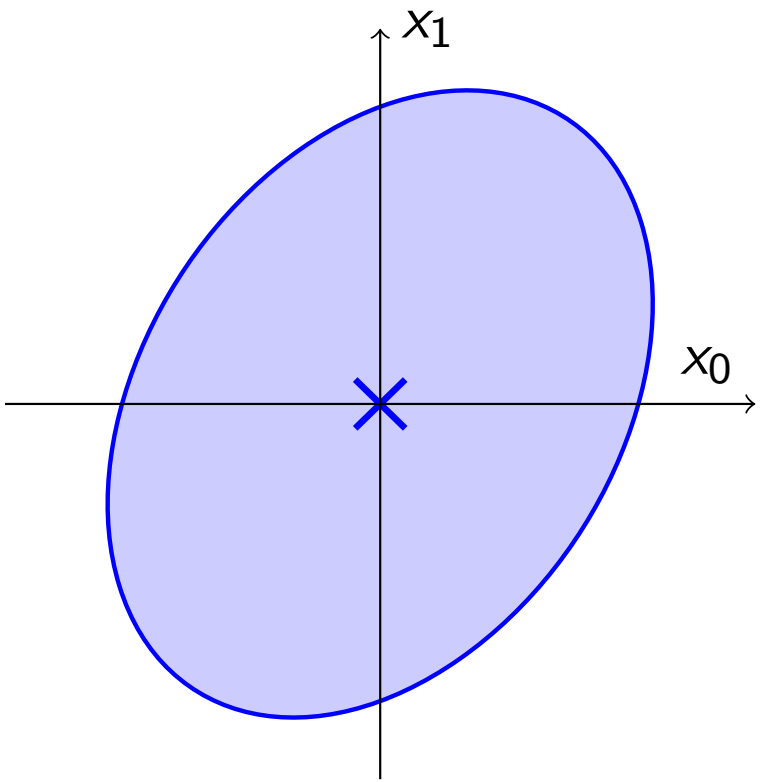
contains initial state
($(0, 0)$ for instance)



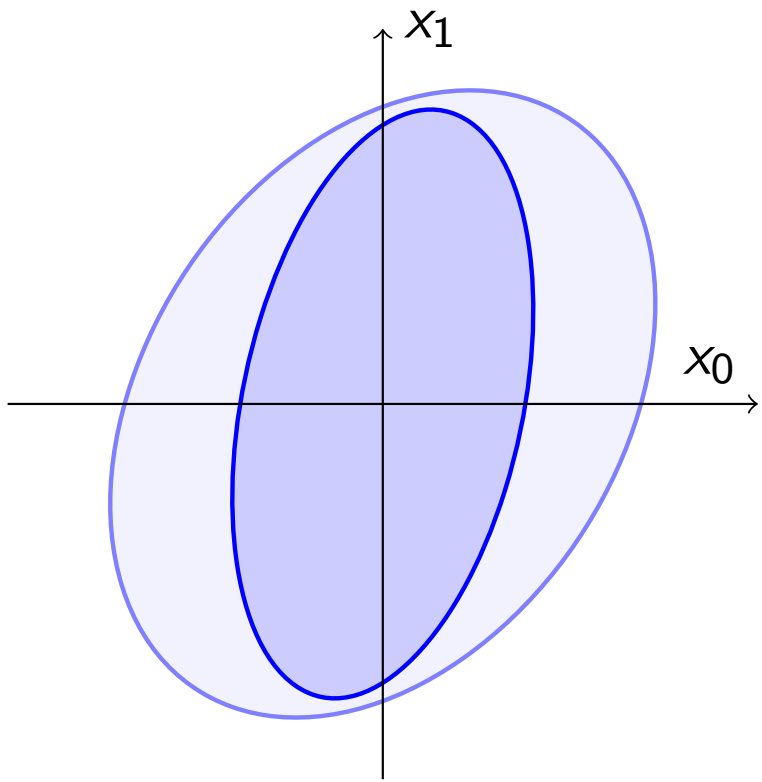
is stable in one step
(loop body)

Inductive Invariants

A set of points is an inductive invariant for a controller if it:



contains initial state
($(0, 0)$ for instance)



is stable in one step
(loop body)

1 Quadratic Invariants

2 Policy Iterations

3 Open Directions

1 Quadratic Invariants

2 Policy Iterations

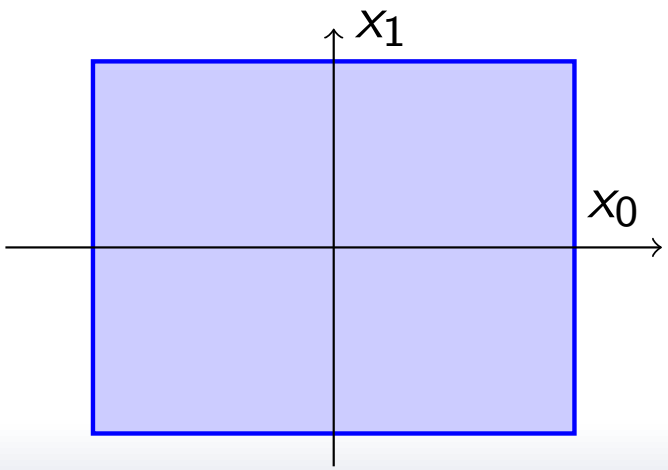
3 Open Directions

Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.

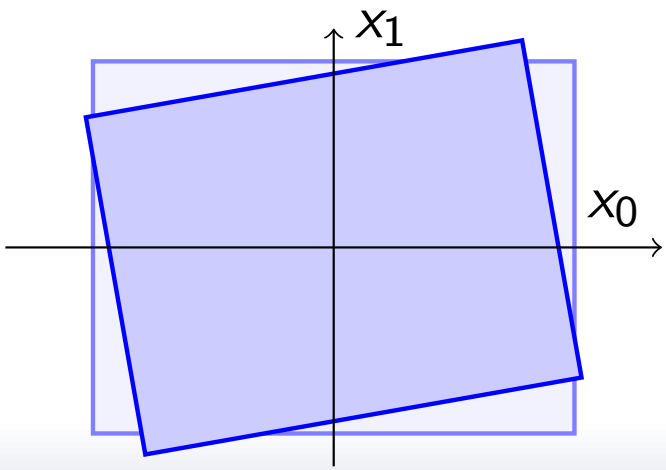
Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



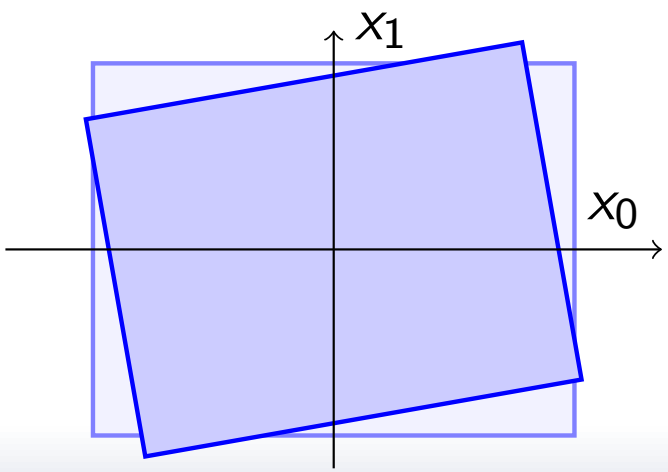
Quadratic invariants

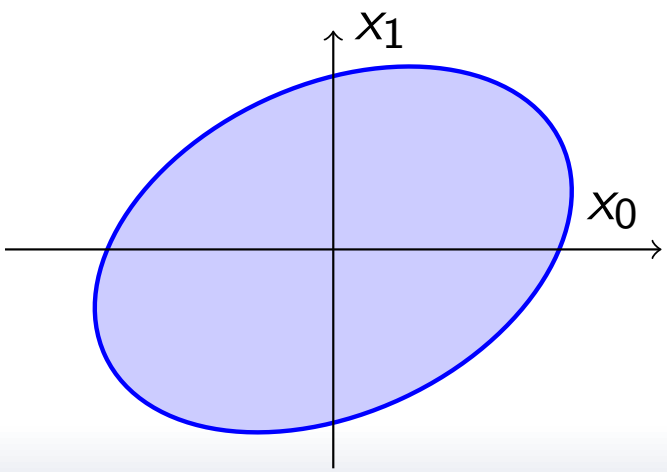
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



Quadratic invariants

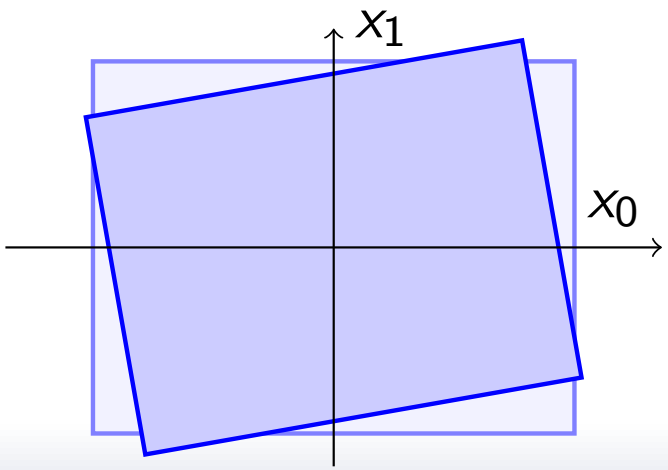
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that **quadratic invariants** are a good fit for linear systems.

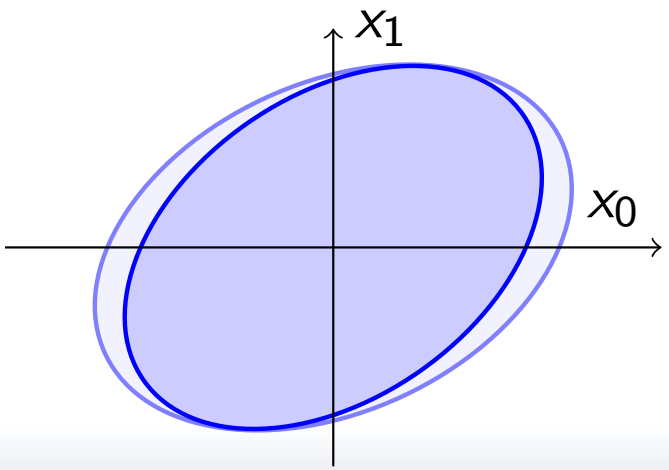




Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that **quadratic invariants** are a good fit for linear systems.





Quadratic Invariants

Remark

Reachable state space is usually **not** an ellipsoid.

Quadratic Invariants

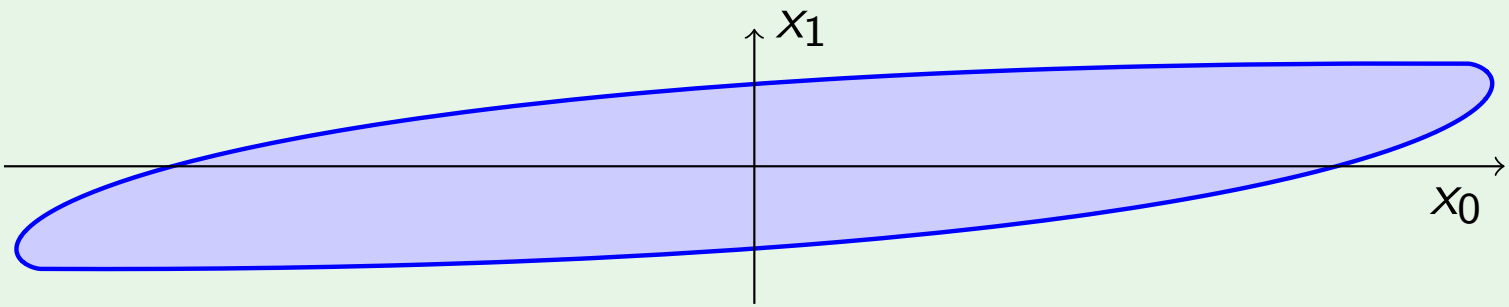
Remark

Reachable state space is usually **not** an ellipsoid.

Example

$x_0 := 0$ and $x_{k+1} := Ax_k + Bu_k$ where $\|u_k\|_\infty \leq 1$ and

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



Quadratic Invariants

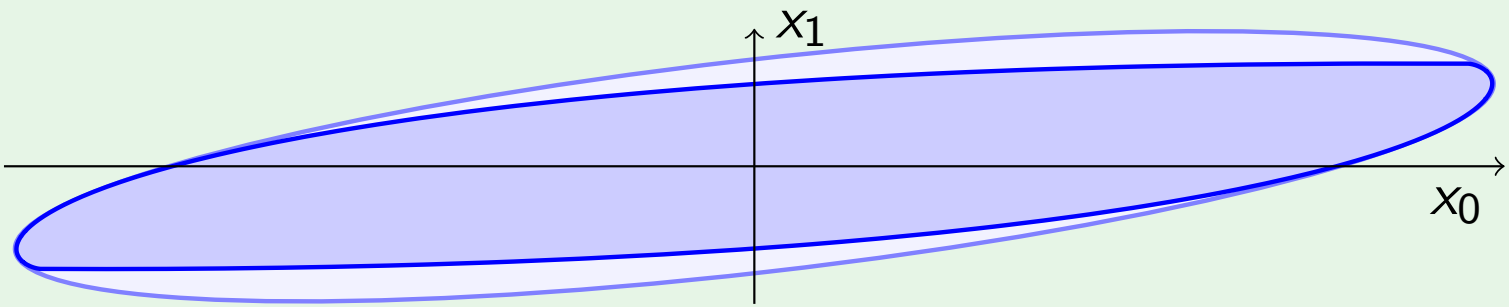
Remark

Reachable state space is usually **not** an ellipsoid.

Example

$x_0 := 0$ and $x_{k+1} := Ax_k + Bu_k$ where $\|u_k\|_\infty \leq 1$ and

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



But remains reasonably close.

Lyapunov Stability [Lyapunov47]

Theorem

For $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, the sequence

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = Ax_k + Bu_k \end{cases}$$

is bounded for all $u \in (\mathbb{R}^p)^{\mathbb{N}}$ such that for all $k \in \mathbb{N}$, $||u_k||_\infty \leq 1$ if and only if there exists $P \in \mathbb{R}^{n \times n}$ positive definite such that

$$P - A^T P A \succ 0$$

where $M \succ 0$ means that for all $x \in \mathbb{R}^n$: $x \neq 0 \Rightarrow x^T M x > 0$.

Lyapunov Stability, Invariant

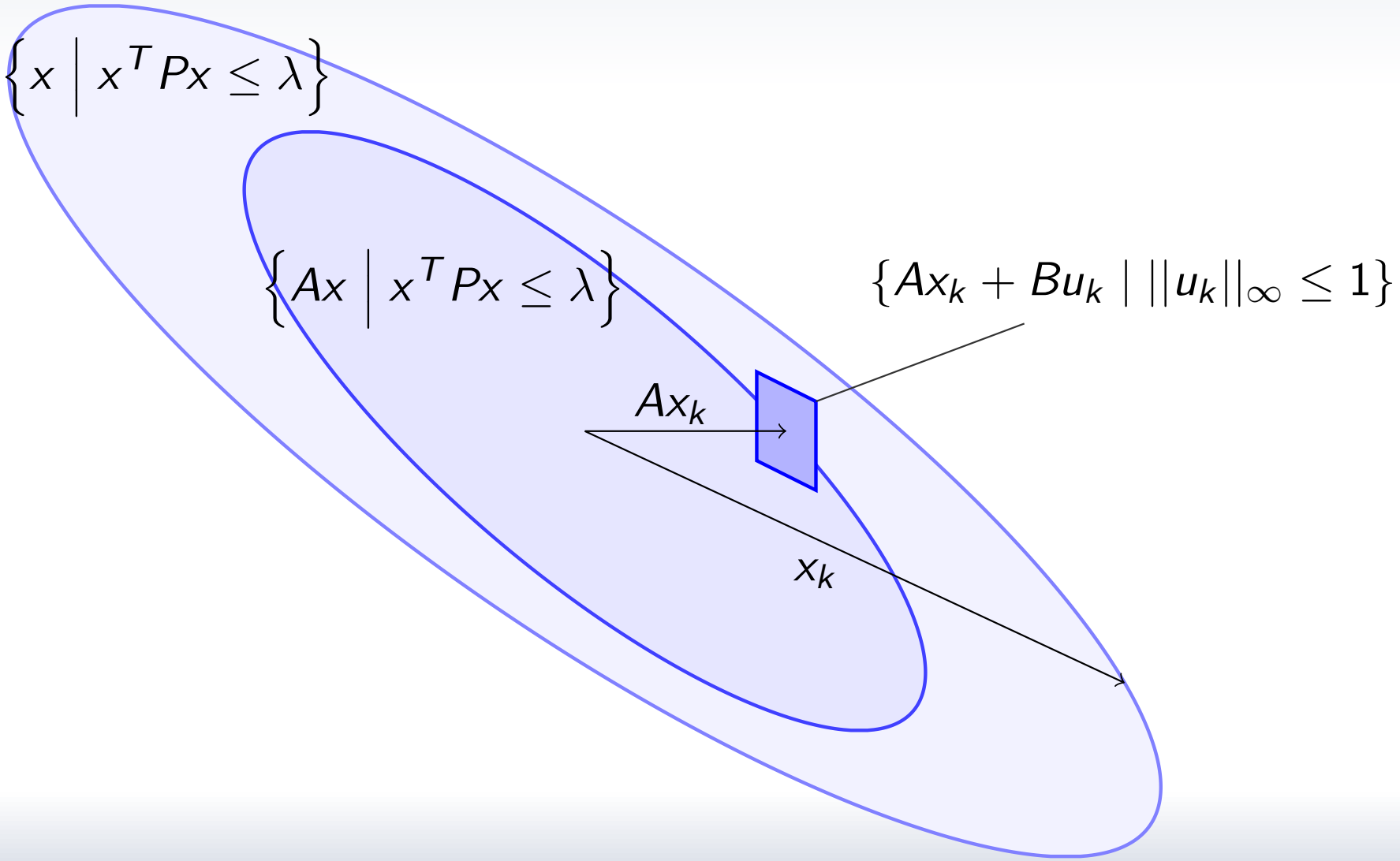
Invariant ellipsoid

Moreover, there exists a $\lambda > 0$ such that
x remains in the ellipsoid $\left\{ x \in \mathbb{R}^n \;\middle|\; x^T P x \leq \lambda \right\}$.

In Computer Science Language

The property "$x^T P x \leq \lambda$" is a **loop invariant**.

Lyapunov Stability, Illustration



Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming [VandenbergheB96]

Minimize a linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

where the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming [VandenbergheB96]

Minimize a linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

where the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

- "Efficient" solvers exist;

Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming [VandenbergheB96]

Minimize a linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

where the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

- "Efficient" solvers exist;
- Unknown matrices: Y can be decomposed as $\sum_{i,j} y_{i,j} E^{i,j}$;

Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming [VandenbergheB96]

Minimize a linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

where the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

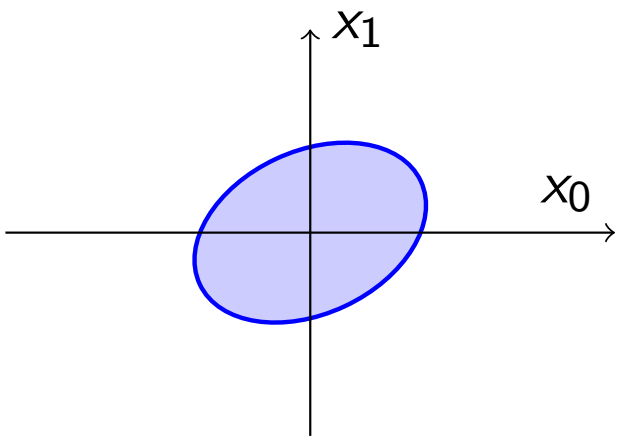
- "Efficient" solvers exist;
- Unknown matrices: Y can be decomposed as $\sum_{i,j} y_{i,j} E^{i,j}$;
- Multiple constraints: $A \succeq 0 \wedge B \succeq 0$ is equivalent to $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \succeq 0$.

Shape of the Ellipsoid

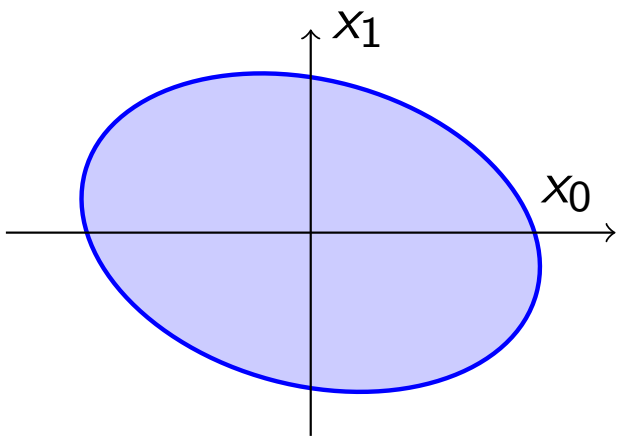
- We look for a positive definite matrix P such that $P - A^T P A \succ 0$.

Shape of the Ellipsoid

- We look for a positive definite matrix P such that $P - A^T P A \succ 0$.
- We will then look for a $\lambda > 0$ such that $x^T P x \leq \lambda$ is an invariant and the ellipsoid $\left\{ x \;\middle|\; x^T P x \leq \lambda \right\}$ should be "small".



is better than

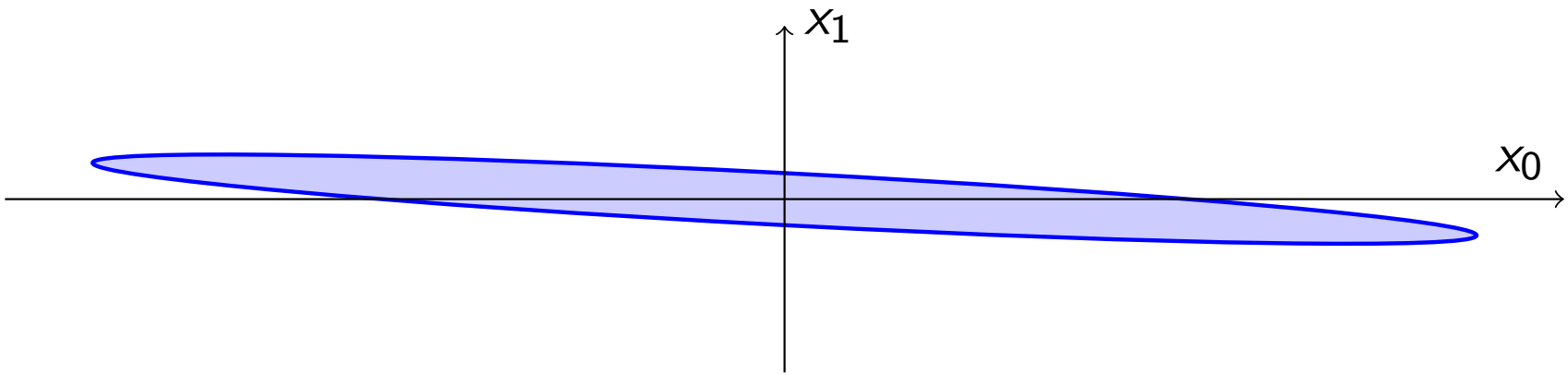


Heuristic 1: Minimizing Condition Number

We can look for the roundest possible ellipsoid
by minimizing a new variable r in the extra constraint

$$I \preceq P \preceq rI.$$

Rationale: "flat" ellipsoids can yield a very bad bound
on one of the variables.

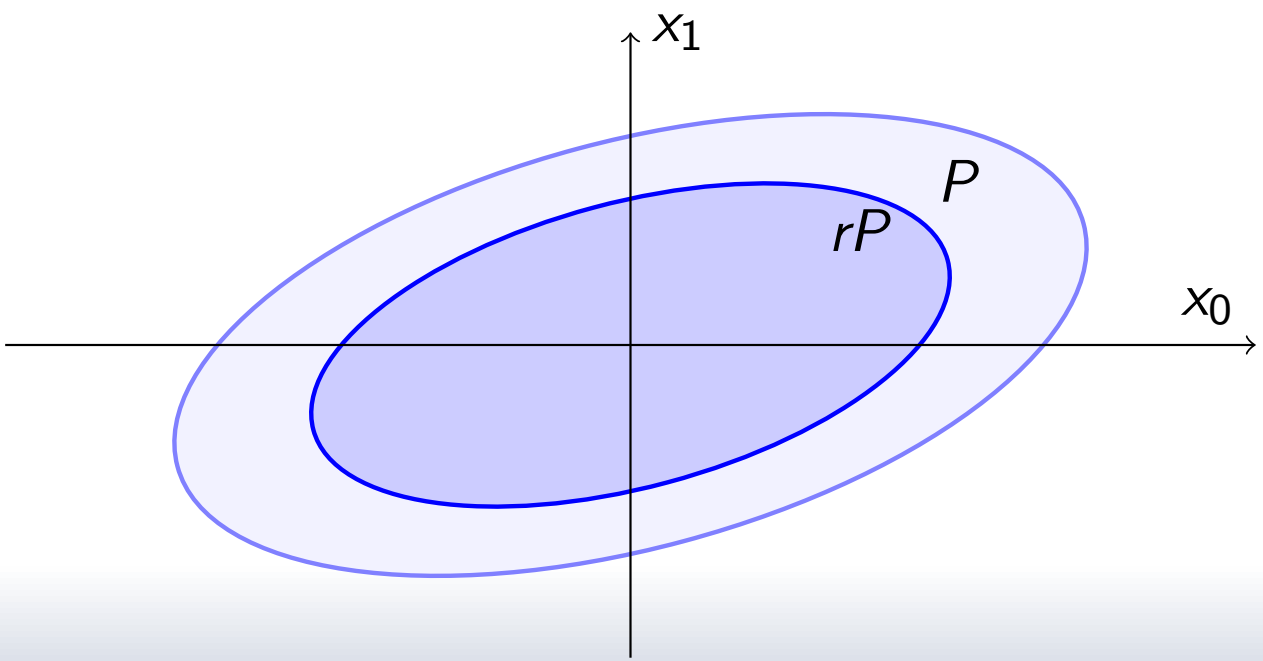


Heuristic 2: Preserving the Shape

Another heuristic is to minimize $r \in (0, 1)$ in

$$rP - A^T P A \succeq 0.$$

Intuitively, the shape of ellipsoid that gets "preserved" the best through the update $x_{k+1} = Ax_k$.

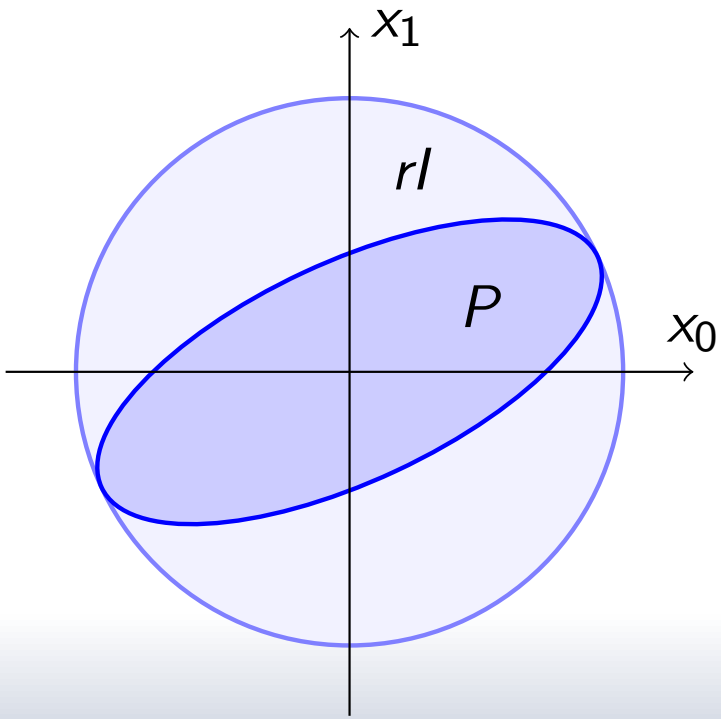


Heuristic 3: All in One

The two previous heuristics did not take B into account.
We could look for P maximizing r in $P \succeq rI$ such that

$$\forall x, \forall u, ||u||_\infty \leq 1 \Rightarrow x^T P x \leq 1 \Rightarrow (Ax + Bu)^T P (Ax + Bu) \leq 1.$$

Intuitively, invariant ellipsoid included in smallest possible sphere.

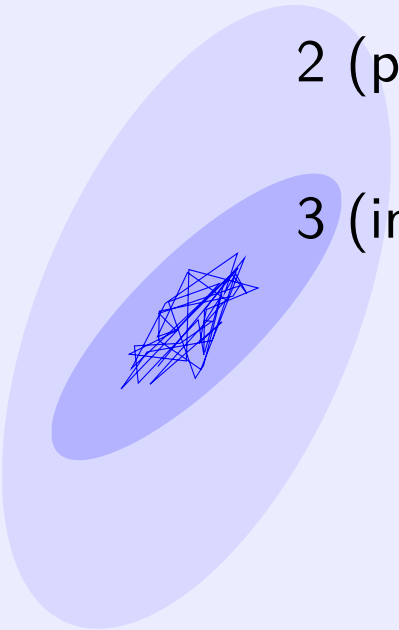


Example

1 (condition number)

2 (preserving shape)

3 (in smallest sphere)



Experimental Results

	H.	Bounds	Reachable
Ex. 2 n=4, 1 input	1	1661, 1795, 843, 1288	1.42, 1.42, 1.00, 1.00
	2	5.50, 6.71, 2.20, 3.44	
	3	1.69, 1.92, 2.13, 2.42	
Ex. 3 Lead-lag controller n=2, 1 input	1	60.93, 60.52	3.97, 20.00
	2	36.55, 35.50	
	3	28.83, 25.85	
Ex. 4 LQG regulator n=3, 1 input	1	1.26, 1.26, 1.26	0.32, 0.24, 0.22
	2	1.21, 1.23, 1.06	
	3	0.68, 0.41, 0.28	
Ex. 5 Coupled mass system n=4, 2 inputs	1	4980, 5061, 4768, 5693	2.79, 2.73, 3.50, 3.30
	2	6.37, 6.20, 6.07, 9.57	
	3	4.97, 4.90, 4.77, 4.63	
Ex. 6 Low-pass filter n=5, 1 input	1	253, 261, 251, 280, 286	1.42, 0.91, 1.44, 1.52, 2.14
	2	3.11, 4.30, 4.15, 8.16, 8.81	
	3	2.21, 1.10, 1.87, 1.98, 2.83	

Experimental Results, Analysis Times

	Heuristic	t_P (s)	t_λ (s)	$t_{valid.}$ (s)	t_{total} (s)
Ex. 2 n=4, 1 input	1	0.10	0.63	0.02	0.75
	2	0.21	0.37	0.01	0.59
	3	0.33	0.22	0.01	0.56
Ex. 3 Discretized lead-lag controller n=2, 1 input	1	0.08	0.47	0.03	0.60
	2	0.13	0.45	0.02	0.60
	3	0.16	0.21	0.02	0.39
Ex. 4 Linear quadratic gaussian regulator n=3, 1 input	1	0.08	0.33	0.02	0.43
	2	0.14	0.29	0.02	0.45
	3	0.16	0.22	0.02	0.40
Ex. 5 Controller for a coupled mass system n=4, 2 inputs	1	0.09	0.76	0.03	0.88
	2	0.17	0.43	0.03	0.63
	3	0.27	0.23	0.03	0.53
Ex. 6 Butterworth low-pass filter n=5, 1 input	1	0.11	0.65	0.03	0.79
	2	0.22	0.37	0.02	0.61
	3	0.56	0.25	0.02	0.83

On an Intel Core2 @ 2.66GHz.

Floating Point Issues

For efficiency, computations made with floating point, hence rounding errors:

Floating Point Issues

For efficiency, computations made with floating point, hence rounding errors:

- In the analyzed controller:

  Using classic ϵ methods [Higham96]:

  Theorem [Roux-PhD13]

  If $x^T P x \leq \lambda$, $\|u\|_\infty \leq 1$ and $(Ax+Bu)^T P (Ax+Bu) \leq \lambda'$, then

  $$\mathrm{fl}(Ax+Bu)^T P\, \mathrm{fl}(Ax+Bu) \leq \left(\sqrt{\lambda'} + \sqrt{\lambda} a + b\right)^2$$

  with a and b (very small) constants computed from A, B and P.

Floating Point Issues

For efficiency, computations made with floating point, hence rounding errors:

- In the analyzed controller:

  Using classic ϵ methods [Higham96]:

  Theorem [Roux-PhD13]

  If $x^T P x \leq \lambda$, $\|u\|_\infty \leq 1$ and $(Ax+Bu)^T P (Ax+Bu) \leq \lambda'$, then

  $$\mathrm{fl}(Ax+Bu)^T P\, \mathrm{fl}(Ax+Bu) \leq \left(\sqrt{\lambda'} + \sqrt{\lambda} a + b\right)^2$$

  with a and b (very small) constants computed from A, B and P.

Floating Point Issues

- In the analyzer:
  - Checking the invariant $\left\{x \;\middle|\; x^T P x \leq \lambda\right\}$ basically amounts to **checking positive definiteness** of following matrix (with appropriate τ and λ_i non negatives):

$$\begin{bmatrix} -A^T PA & -A^T PB & 0 \\ -B^T PA & -B^T PB & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \tau \begin{bmatrix} -P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \sum_{i=0}^{p-1} \lambda_i \begin{bmatrix} 0 & 0 & 0 \\ 0 & -E^{i,i} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

  - This is done by carefully bounding the rounding errors in a **Cholesky decomposition** [Rump06].
  - Hence an efficient soundness check (in $\mathrm{O}\left(n^3\right)$ floating point operations for an $n \times n$ matrix).

Summary

State of the Art (Static Analysis of Controllers)

- Linear Domains [Halbwachs78,...]
- Ellipsoids of dimension 2 [Feret04]
- Unrolling (no inductive invariant) [Feret04, Khalil09,...]

Contributions [RouxJGF-HSCC12]

Automatic computation of Lyapunov functions:

- Heuristics to optimize precision of invariants.
- Overapproximation of floating point computations.
- Soundness check.

1 Quadratic Invariants

2 Policy Iterations

3 Open Directions

Linear Systems with Guards

Actual systems are not purely linear but contain **guards**: resets, saturations, anti-windups,. . .

Linear Systems with Guards

Actual systems are not purely linear but contain **guards**: resets, saturations, anti-windups,...

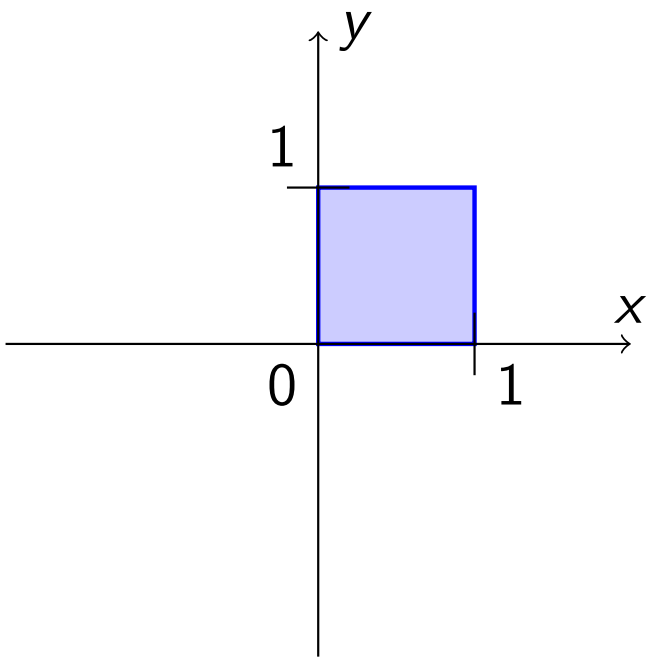
Example

$x := ?(0, 1)$; $y := ?(0, 1)$;
while $-1 \leq 0$ **do**
  $in := ?(0, 1)$;
  if $in \geq 0.9$ **then**
    $x := 10 \times in - 9$;
    $y := 10 \times in - 9$
  else
    $t := x$;
    $x := 0.2 \times t - 0.7 \times y + 0.5 \times in$;
    $y := 0.7 \times t + 0.2 \times y + 0.5 \times in$
  fi
od

Kleene Iterations

Classic abstract interpretation with interval domain:

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```

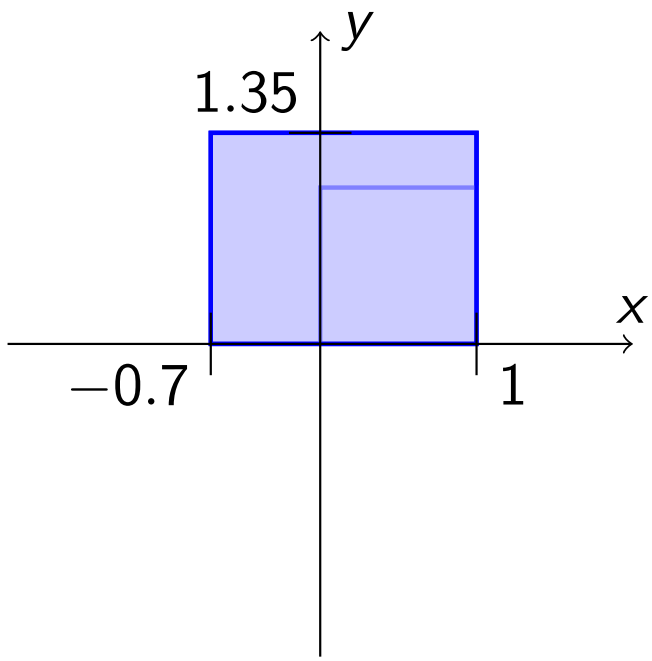


before entering the loop

Kleene Iterations

Classic abstract interpretation with interval domain:

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```

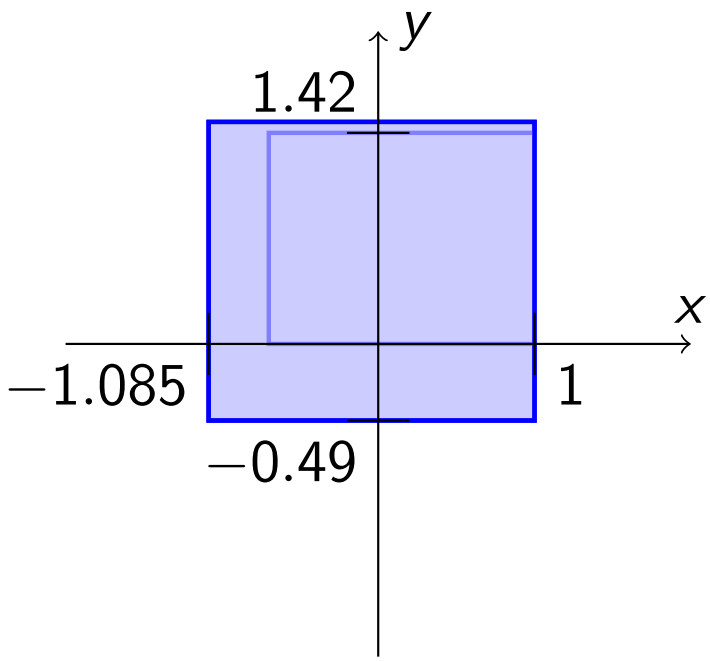


after a first iteration

Kleene Iterations

Classic abstract interpretation with interval domain:

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```

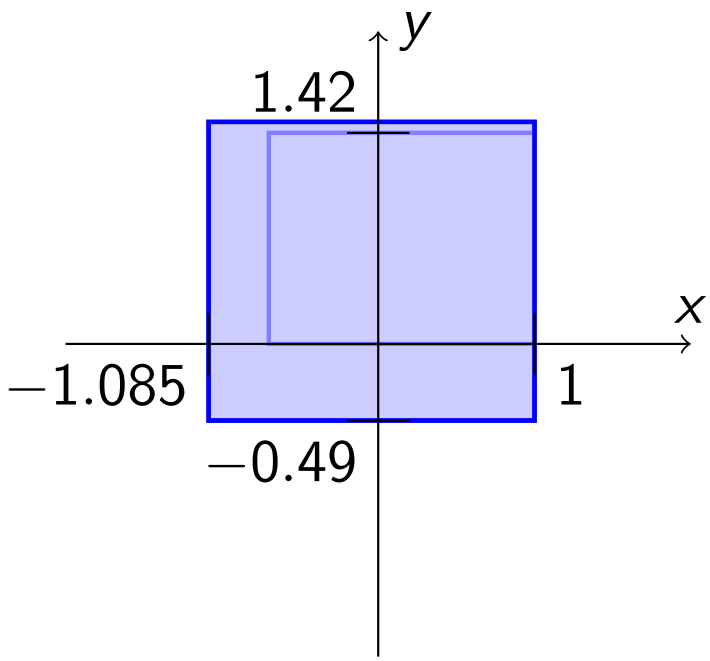


after a second iteration

Kleene Iterations

Classic abstract interpretation with interval domain:

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



. . . does not converge

Widening for Kleene Iterations

Basic Idea

Jump to ensure convergence in **finitely many** iterations.

Example (Simplest Widening)

Jump to $(-\infty, +\infty)$ as soon as bounds are not stable.

Widening for Kleene Iterations

Basic Idea

Jump to ensure convergence in **finitely many** iterations.

Example (Simplest Widening)

Jump to $(-\infty, +\infty)$ as soon as bounds are not stable.

Leads to $x \in (-\infty, +\infty) \wedge y \in (-\infty, +\infty)$

Widening for Kleene Iterations

Basic Idea

Jump to ensure convergence in **finitely many** iterations.

Example (Simplest Widening)

Jump to $(-\infty, +\infty)$ as soon as bounds are not stable.

Leads to $x \in (-\infty, +\infty) \wedge y \in (-\infty, +\infty)$: useless result.

Widening for Kleene Iterations

Basic Idea

Jump to ensure convergence in **finitely many** iterations.

Example (Simplest Widening)

Jump to $(-\infty, +\infty)$ as soon as bounds are not stable.

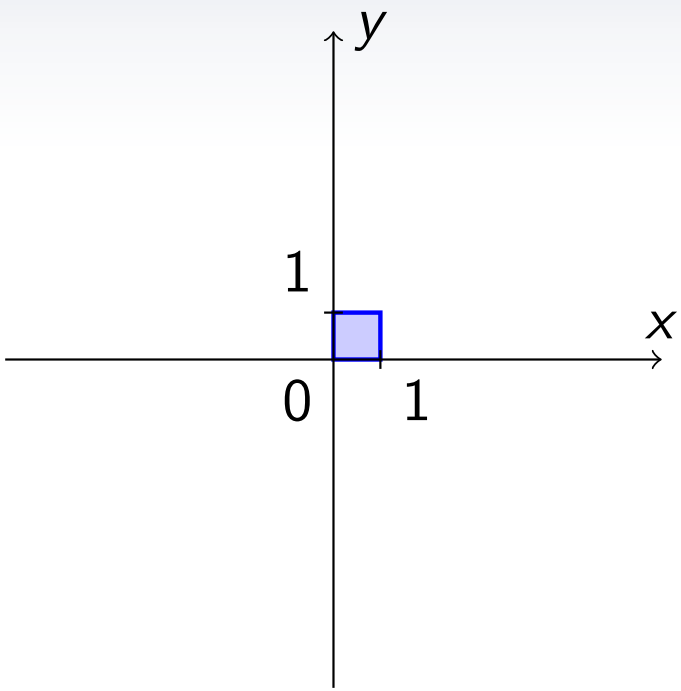
Leads to $x \in (-\infty, +\infty) \wedge y \in (-\infty, +\infty)$: useless result.

Example (Widening with Thresholds)

Stop on a finite number of thresholds T on the way to infinity.
For instance $T = \{ -5, 5 \}$.

Widening for Kleene Iterations

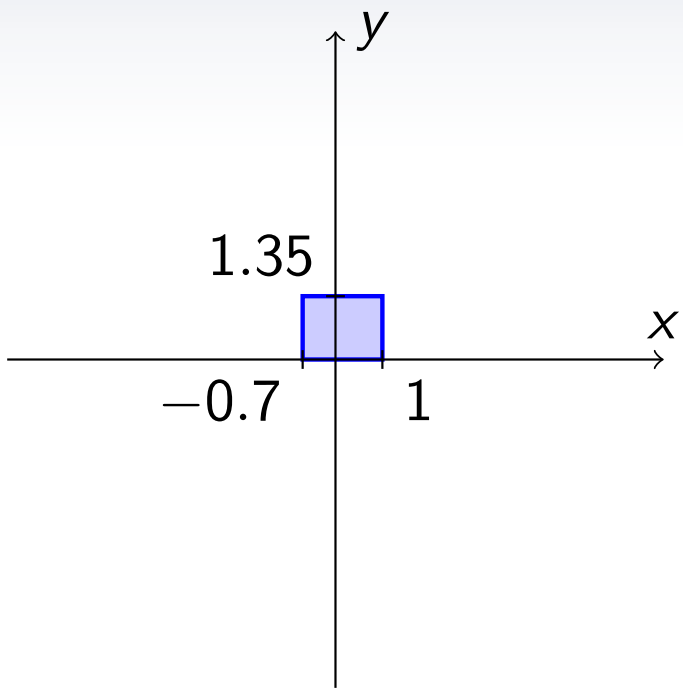
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



before entering the loop

Widening for Kleene Iterations

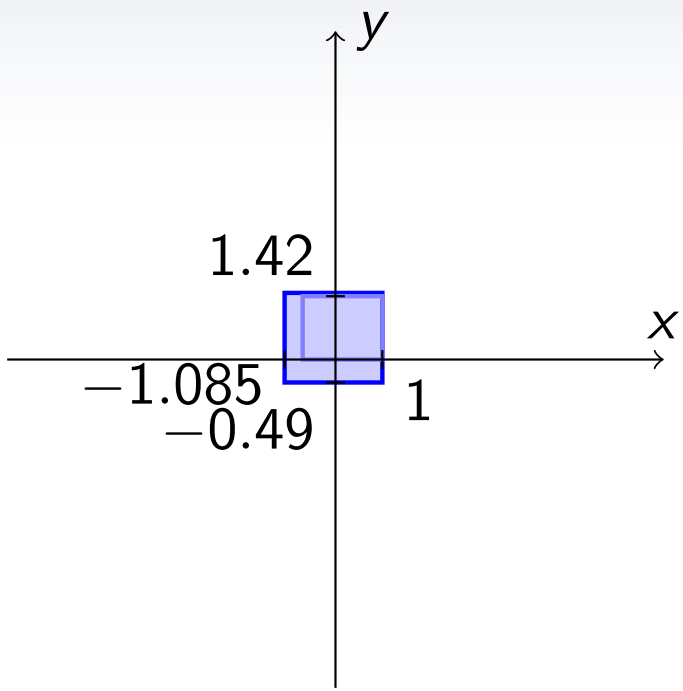
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after a first iteration

Widening for Kleene Iterations

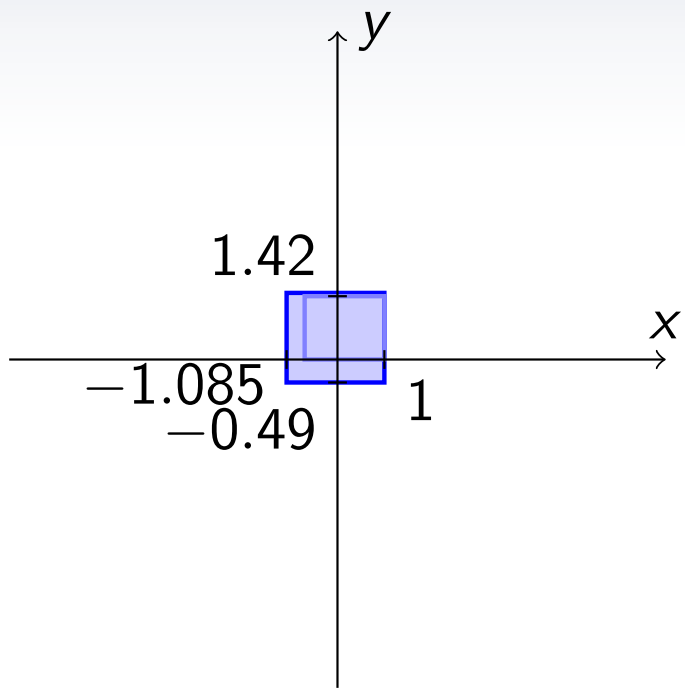
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
  in := ?(0, 1);
  if 0.9 − in ≤ 0 then
    x := 10×in − 9;
    y := 10×in − 9
  else
    t := x;
    x := 0.2×t − 0.7×y + 0.5×in;
    y := 0.7×t + 0.2×y + 0.5×in
  fi
od
```



after a second iteration

Widening for Kleene Iterations

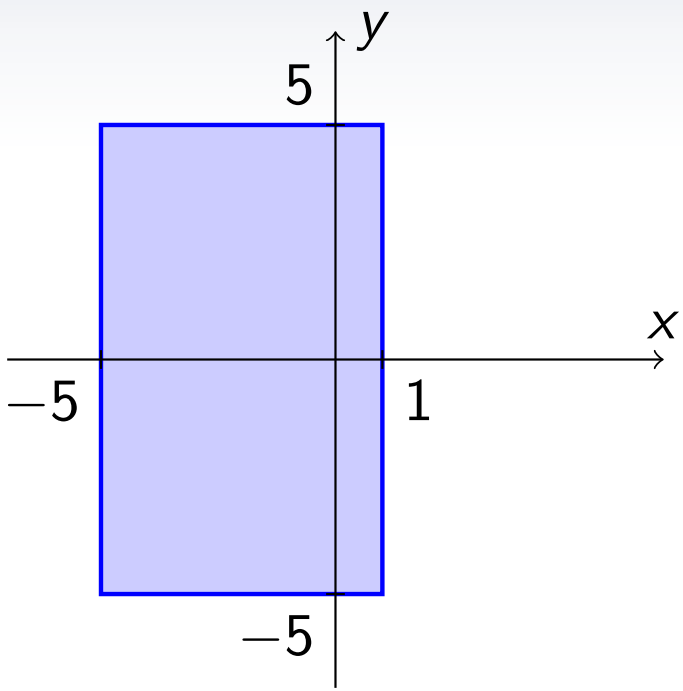
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
  in := ?(0, 1);
  if 0.9 − in ≤ 0 then
    x := 10×in − 9;
    y := 10×in − 9
  else
    t := x;
    x := 0.2×t − 0.7×y + 0.5×in;
    y := 0.7×t + 0.2×y + 0.5×in
  fi
od
```



not stable → widening

Widening for Kleene Iterations

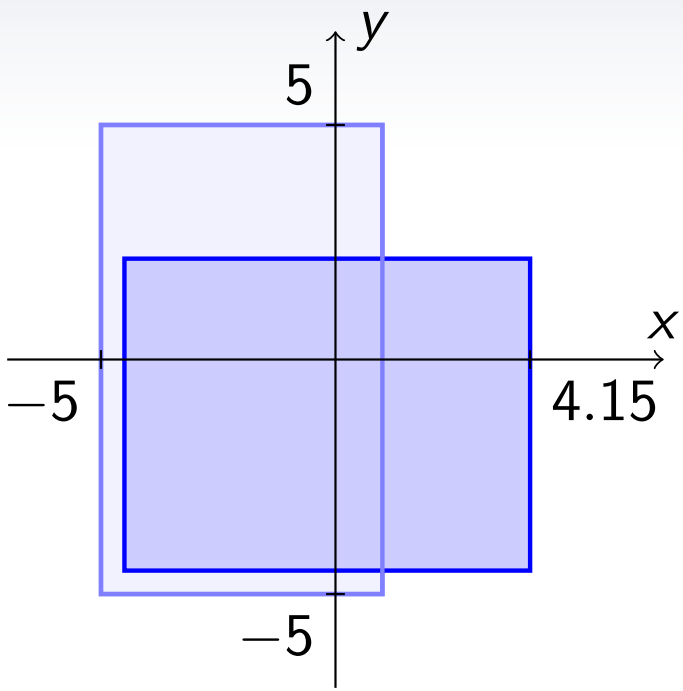
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after widening

Widening for Kleene Iterations

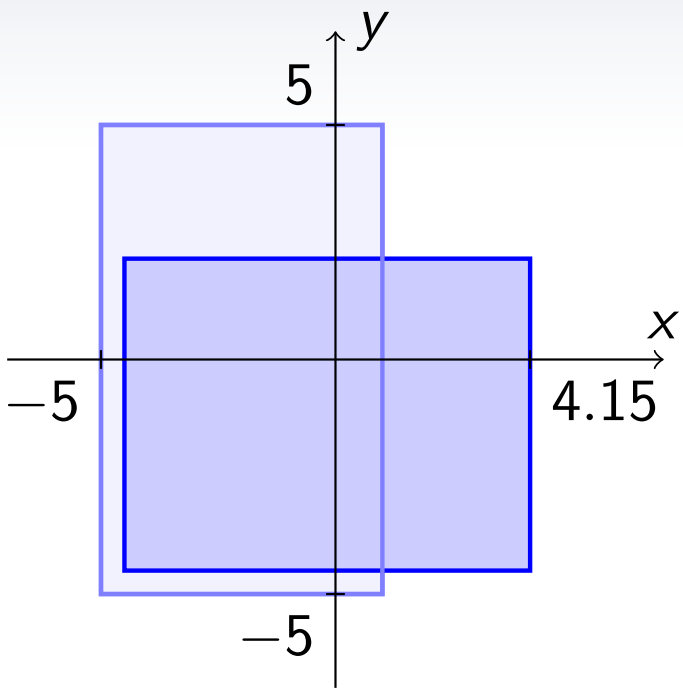
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after another iteration

Widening for Kleene Iterations

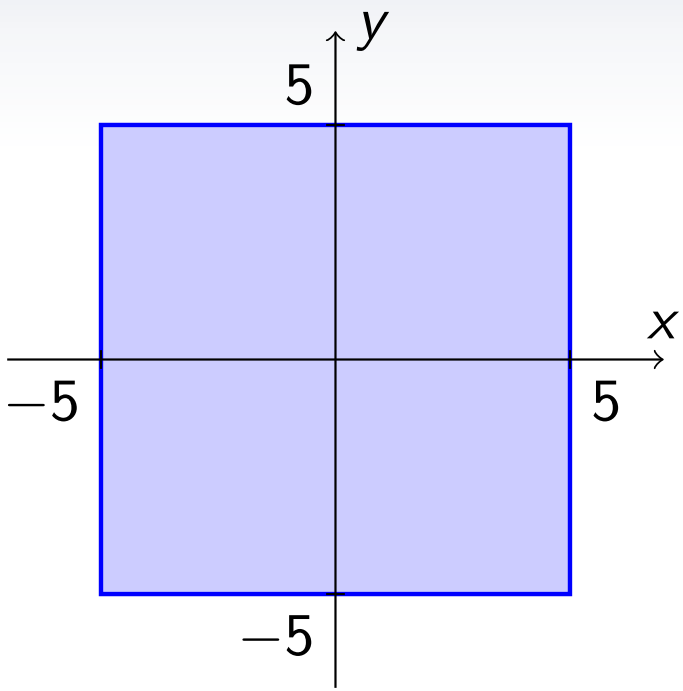
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



not stable → widening

Widening for Kleene Iterations

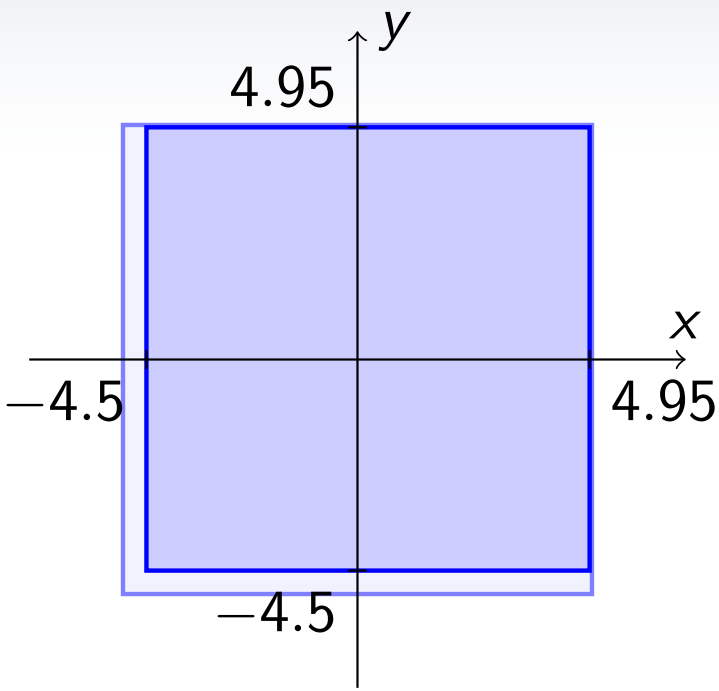
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after widening

Widening for Kleene Iterations

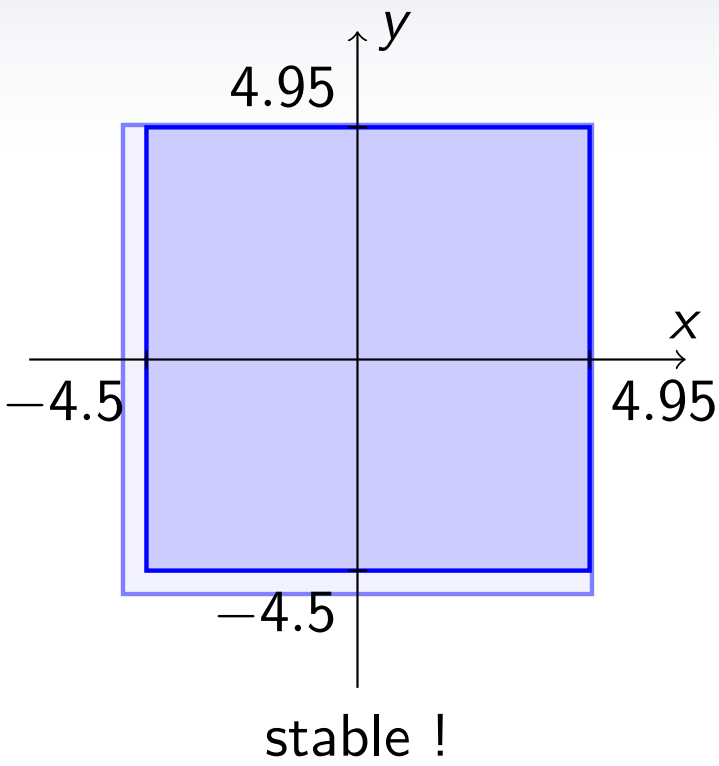
$x := ?(0, 1)$; $y := ?(0, 1)$;
while $-1 \leq 0$ **do**
  in $:= ?(0, 1)$;
  if $0.9 - \text{in} \leq 0$ **then**
    $x := 10 \times \text{in} - 9$;
    $y := 10 \times \text{in} - 9$
  else
    $t := x$;
    $x := 0.2 \times t - 0.7 \times y + 0.5 \times \text{in}$;
    $y := 0.7 \times t + 0.2 \times y + 0.5 \times \text{in}$
  fi
od



after another iteration

Widening for Kleene Iterations

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```

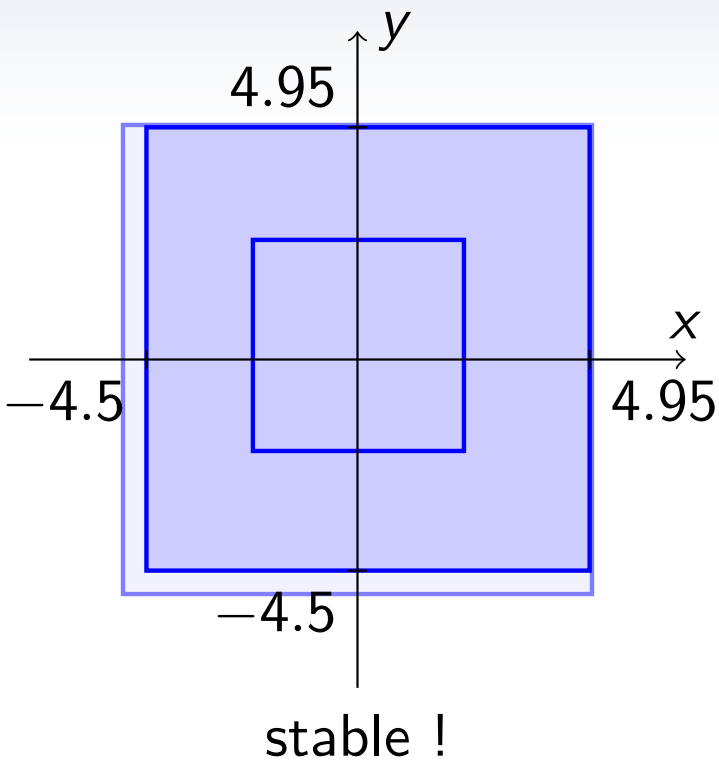


Remarks

- Worthwhile result: $x \in [-5, 5] \wedge y \in [-5, 5]$.

Widening for Kleene Iterations

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```

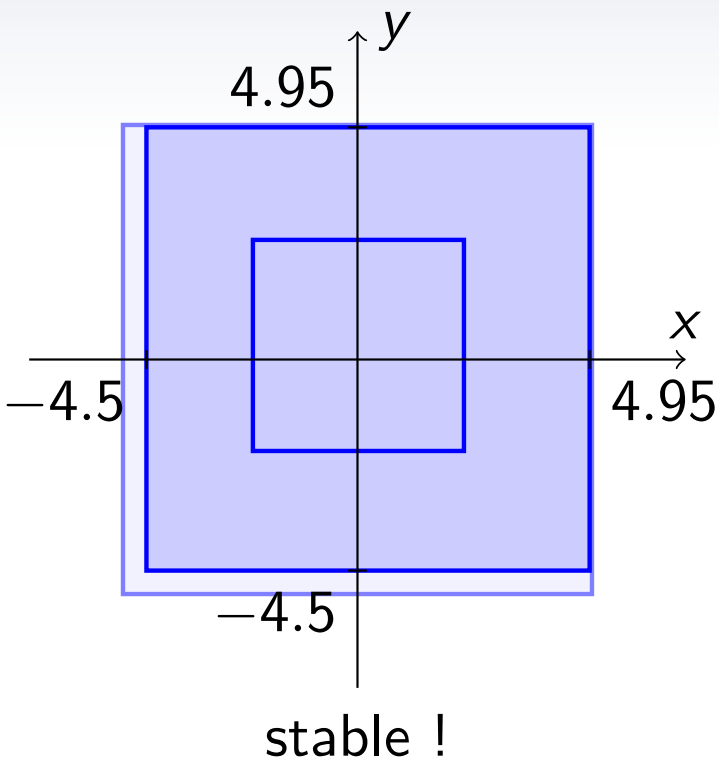


Remarks

- Worthwhile result: $x \in [-5, 5] \wedge y \in [-5, 5]$.
- Larger than least fixpoint: $x \in [-2.23, 2.27] \wedge y \in [-1.95, 2.55]$.

Widening for Kleene Iterations

```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



Remarks

- Worthwhile result: $x \in [-5, 5] \wedge y \in [-5, 5]$.
- Larger than least fixpoint: $x \in [-2.23, 2.27] \wedge y \in [-1.95, 2.55]$.
- Hardly refinable by narrowing.

Policy Iterations

Basic Idea

Use **numerical optimization tools** to compute bounds that are hard to guess for widening.

Policy Iterations

Basic Idea

Use **numerical optimization tools** to compute bounds that are hard to guess for widening.

Example

- Linear programming.
- Semidefinite programming.

Policy Iterations

Basic Idea

Use **numerical optimization tools** to compute bounds that are hard to guess for widening.

Example

- Linear programming.
- Semidefinite programming.

Two major approaches:

- Min-policies [GaubertGTZ07, AdjéGG10, GawlitzaSAGG12].
- Max-policies [GawlitzaS07, GawlitzaS10, GawlitzaSAGG12].

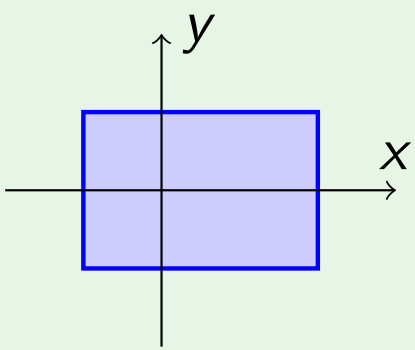
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

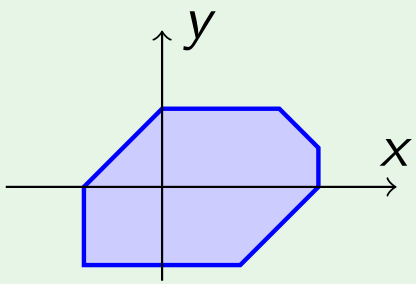
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

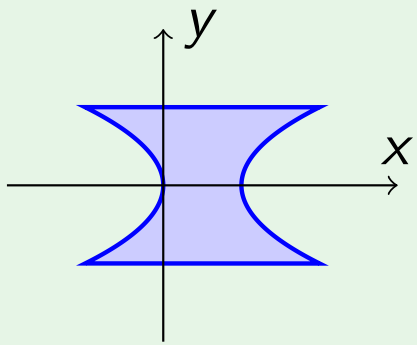
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

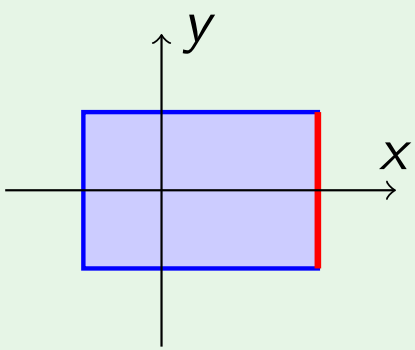
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

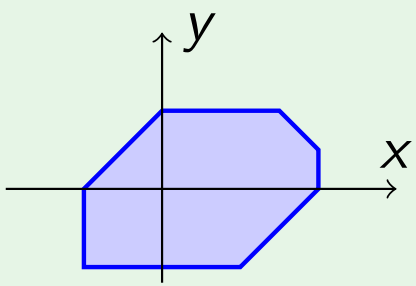
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

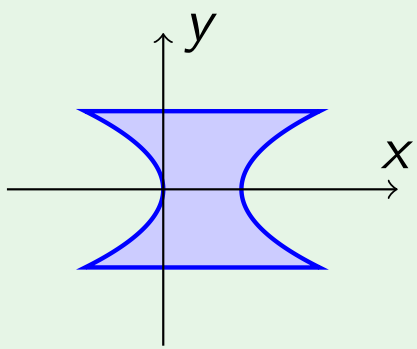
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

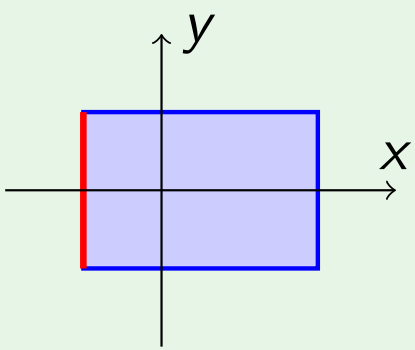
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

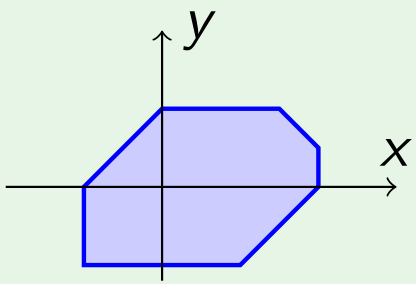
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

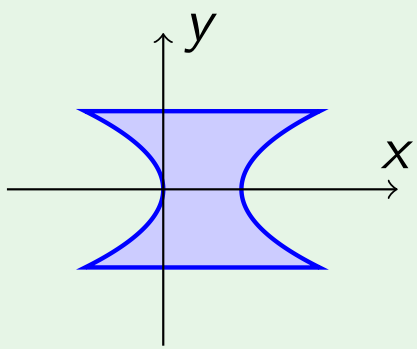
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

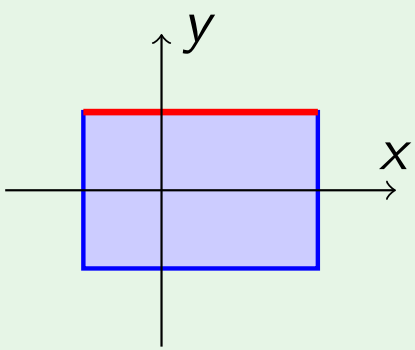
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

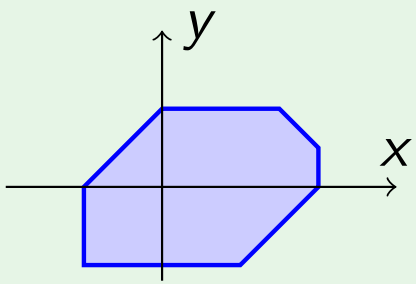
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

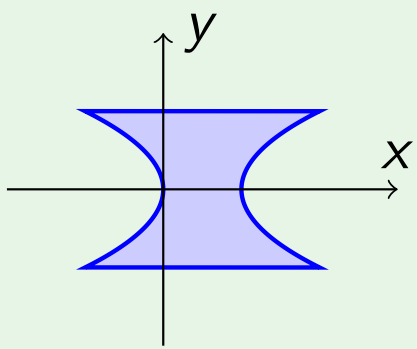
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

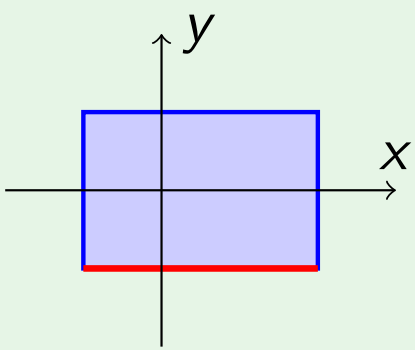
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

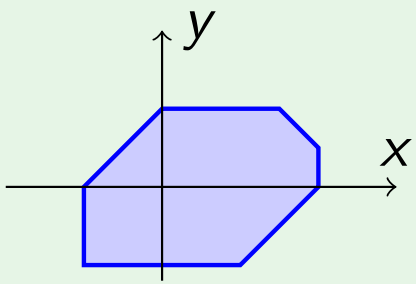
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

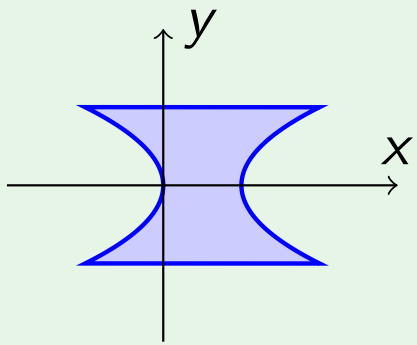
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

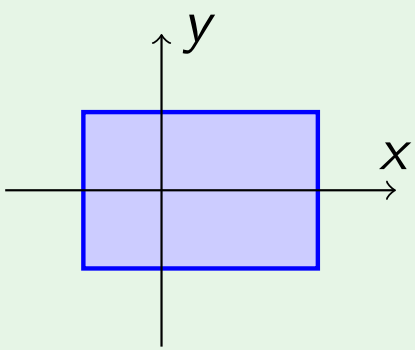
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

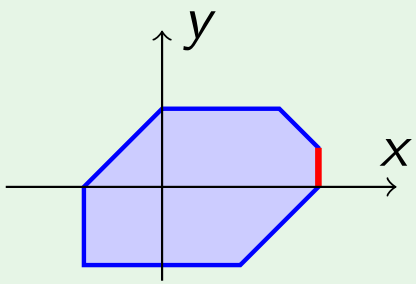
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

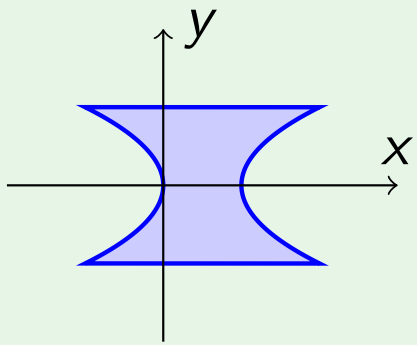
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

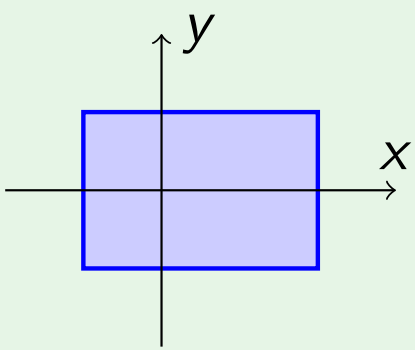
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

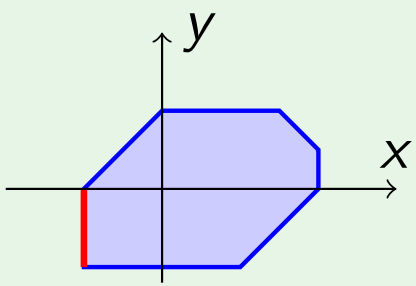
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

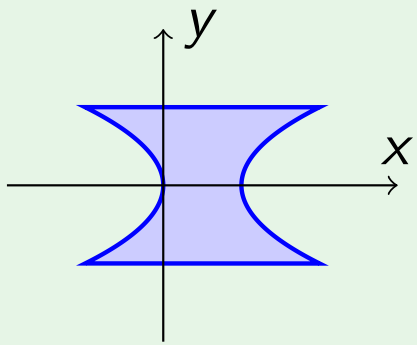
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

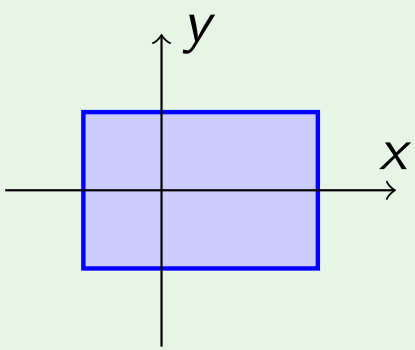
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

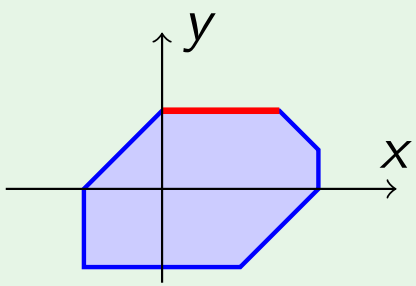
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

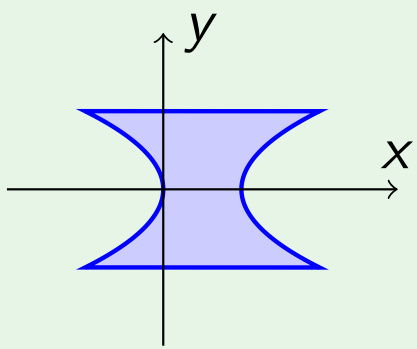
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

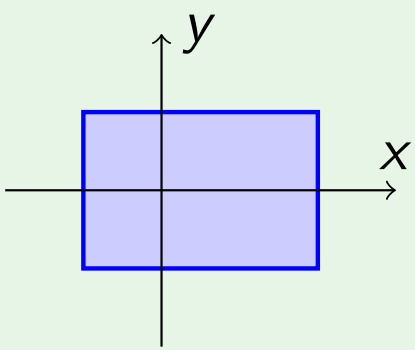
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

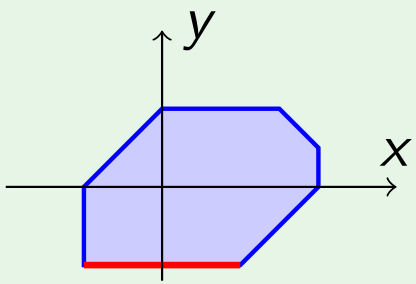
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

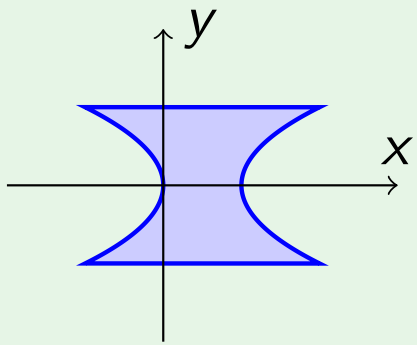
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

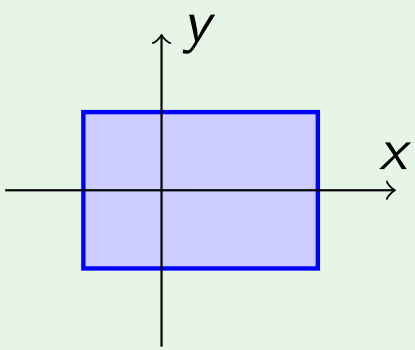
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

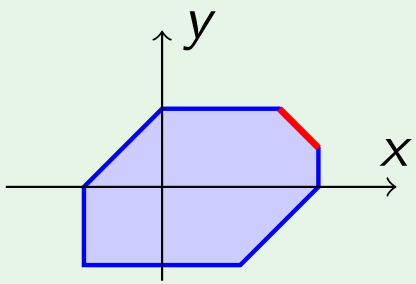
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

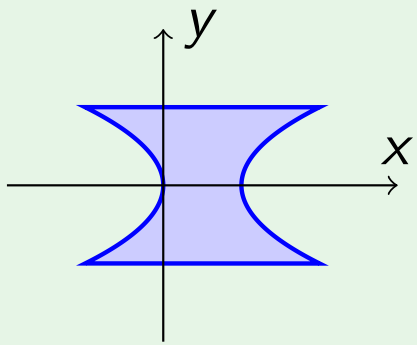
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

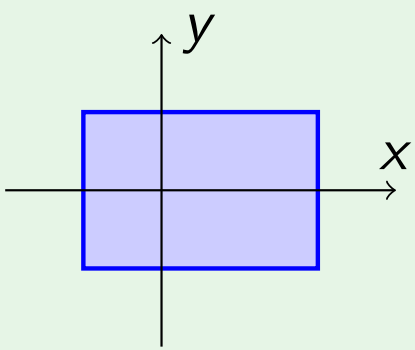
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

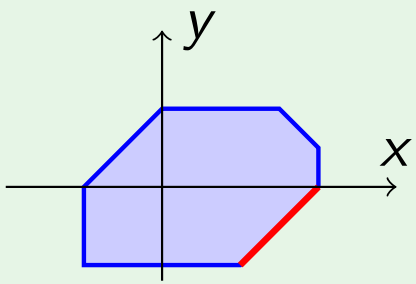
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

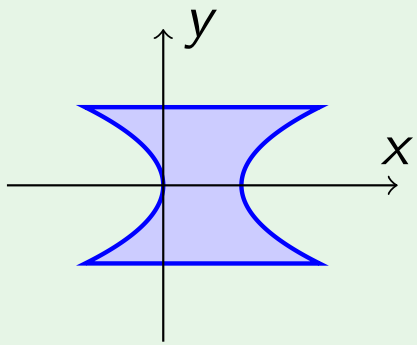
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

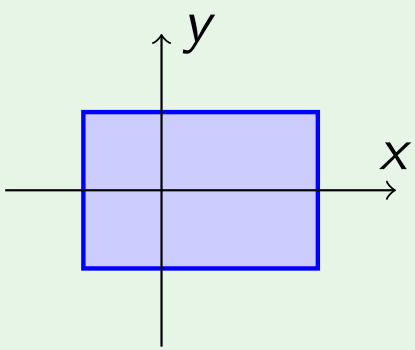
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

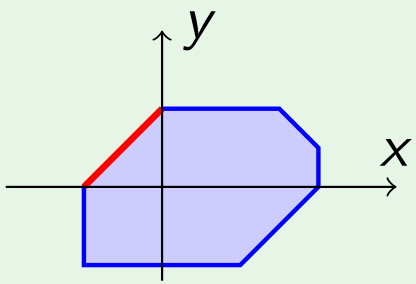
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

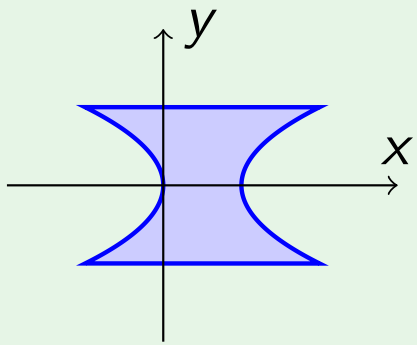
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

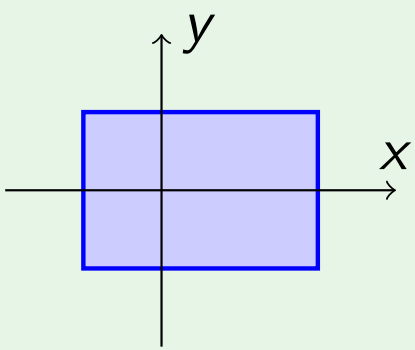
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

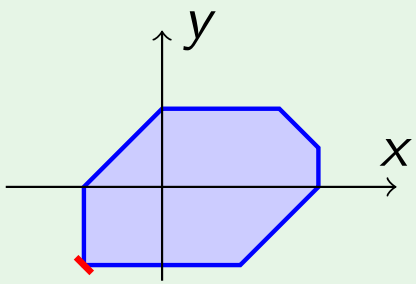
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

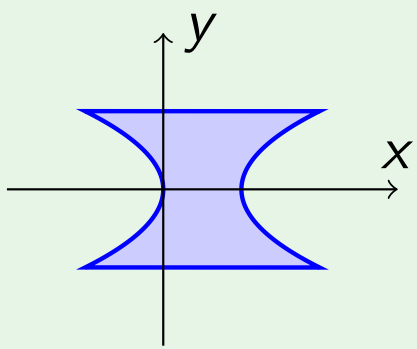
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

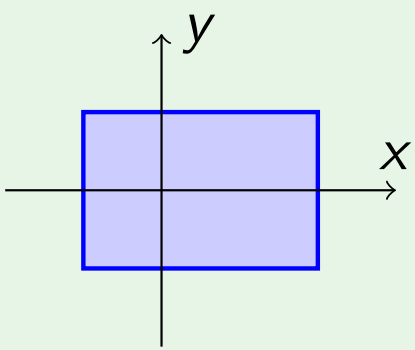
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

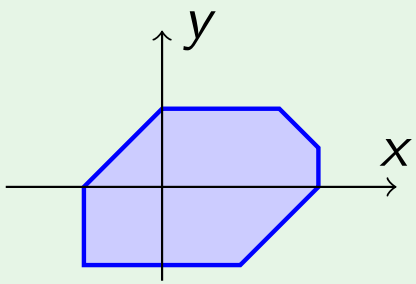
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

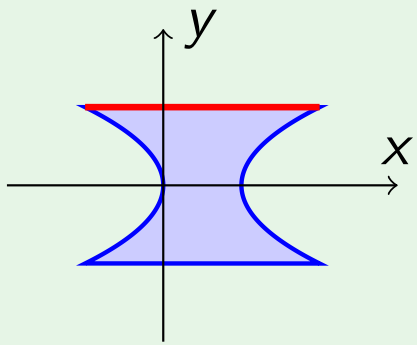
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

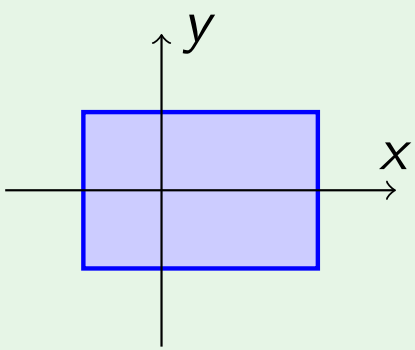
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

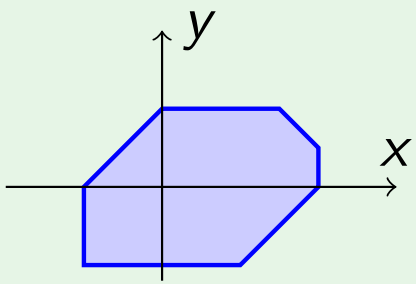
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

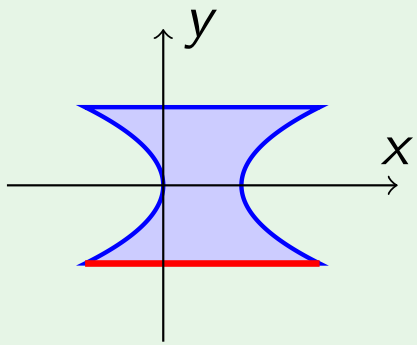
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

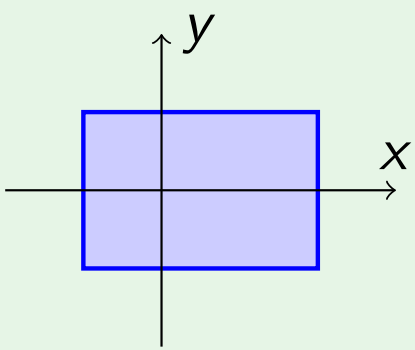
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

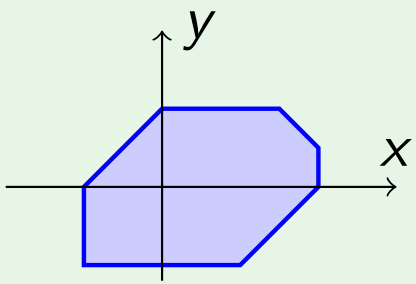
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

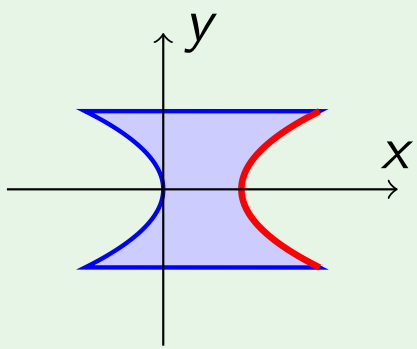
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

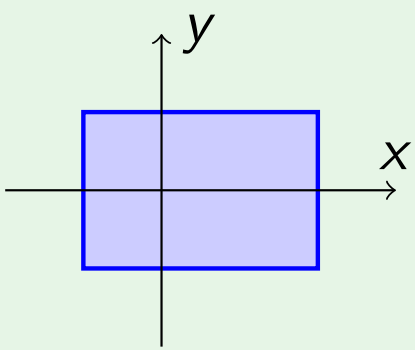
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

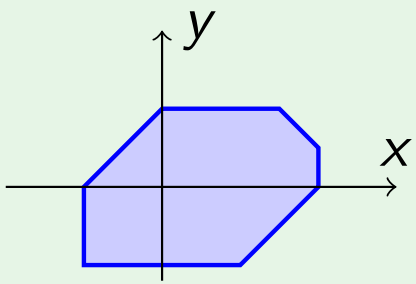
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

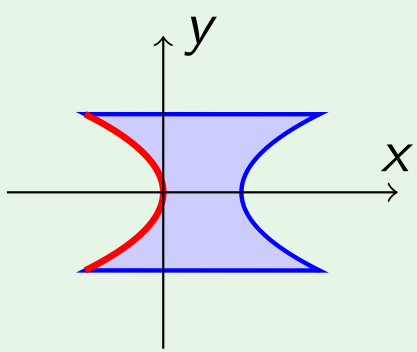
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$



quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

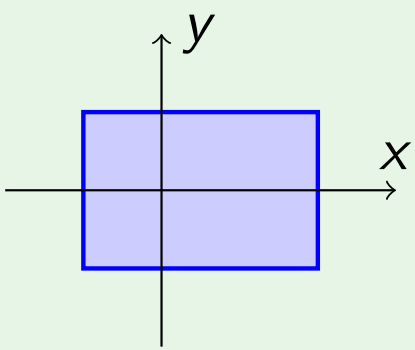
Template Domains

Definition (Template Domain)

Given a set of expressions $\mathcal{T} = \{ t_1, \ldots, t_n \}$, abstract values are tuples $(b_1, \ldots, b_n) \in \overline{\mathbb{R}}^n = (\mathbb{R} \cup \{ \pm\infty \})^n$, representing $\gamma_{\mathcal{T}}(b_1, \ldots, b_n) = \{\rho \in (\mathbb{V} \to \mathbb{R}) \mid t_1(\rho) \leq b_1, \ldots, t_n(\rho) \leq b_n\}$.

Example

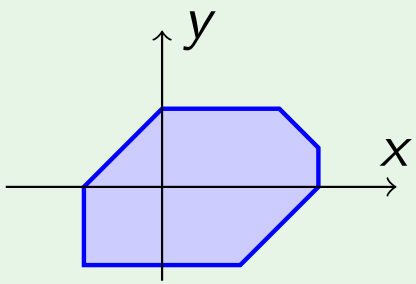
$(2, 1, 1, 1)$



intervals

$(\mathcal{T} = \{ x, -x, y, -y \})$

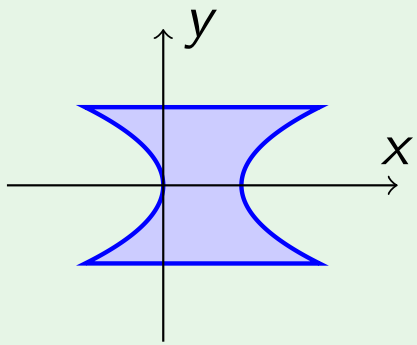
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

$$(\mathcal{T} = \left\{ \begin{array}{c} x, -x, y, -y, \\ x+y, x-y, \\ -x+y, -x-y \end{array} \right\})$$

$(1, 1, 1, 0)$

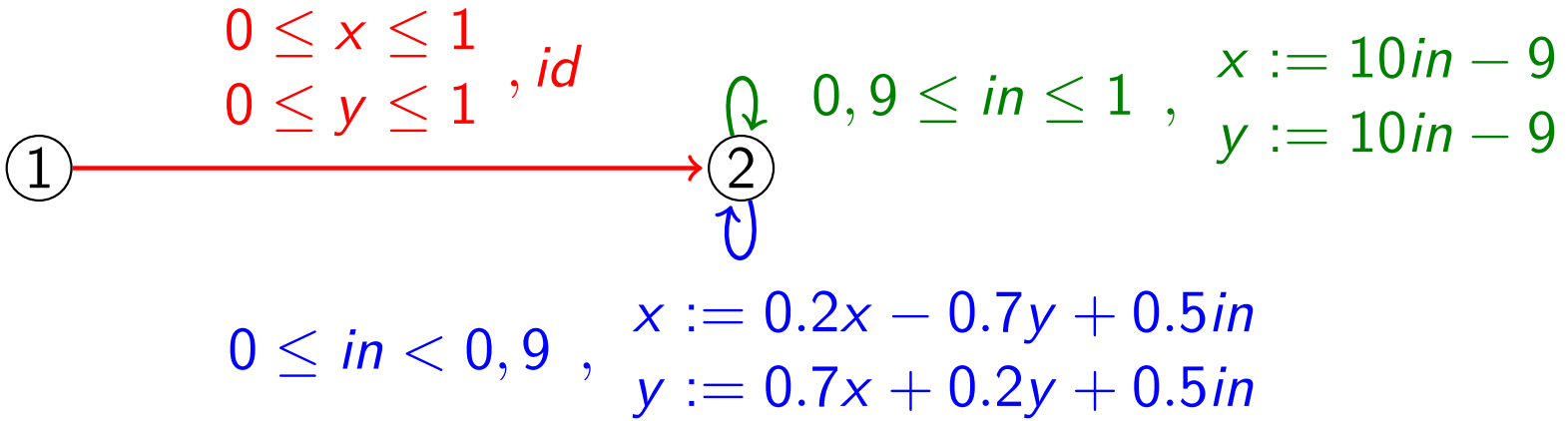


quadratic

$(\mathcal{T} = \{ y, -y, x - y^2, -y^2 - x \})$

System of Equations

First step: building the **control flow graph**.



System of Equations

Second step: deriving a system of equations
from the graph and the templates ($\mathcal{T} = \{ x, -x, y, -y \}$).

$$
\begin{cases}
b_{1,1} = +\infty \qquad b_{1,2} = +\infty \qquad b_{1,3} = +\infty \qquad b_{1,4} = +\infty \\
b_{2,1} = \max\{x \mid 0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \wedge \mathrm{be}(1)\} \\
\qquad \vee \max\{10\, in - 9 \mid 0.9 \leq in \leq 1 \wedge \mathrm{be}(2)\} \\
\qquad \vee \max\{0.2\, x - 0.7\, y + 0.5\, in \mid 0 \leq in \leq 0.9 \wedge \mathrm{be}(2)\} \\
b_{2,2} = \max\{-x \mid 0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \wedge \mathrm{be}(1)\} \\
\qquad \vee \max\{-10\, in + 9 \mid 0.9 \leq in \leq 1 \wedge \mathrm{be}(2)\} \\
\qquad \vee \max\{-0.2\, x + 0.7\, y - 0.5\, in \mid 0 \leq in \leq 0.9 \wedge \mathrm{be}(2)\} \\
b_{2,3} = \max\{y \mid 0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \wedge \mathrm{be}(1)\} \\
\qquad \vee \max\{10\, in - 9 \mid 0.9 \leq in \leq 1 \wedge \mathrm{be}(2)\} \\
\qquad \vee \max\{0.7\, x + 0.2\, y + 0.5\, in \mid 0 \leq in \leq 0.9 \wedge \mathrm{be}(2)\} \\
b_{2,4} = \max\{-y \mid 0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \wedge \mathrm{be}(1)\} \\
\qquad \vee \max\{-10\, in + 9 \mid 0.9 \leq in \leq 1 \wedge \mathrm{be}(2)\} \\
\qquad \vee \max\{-0.7\, x - 0.2\, y - 0.5\, in \mid 0 \leq in \leq 0.9 \wedge \mathrm{be}(2)\}
\end{cases}
$$

with $\mathrm{be}(i)$ denoting $x \leq b_{i,1} \wedge -x \leq b_{i,2} \wedge y \leq b_{i,3} \wedge -y \leq b_{i,4}$

Solving the Equations

1. Decompose the system in **policies** (a.k.a., strategies);
2. Solve them using a linear programming solver.

Solving the Equations

1. Decompose the system in **policies** (a.k.a., strategies);
2. Solve them using a linear programming solver.

Example

Hence $(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) = (2.27, 2.23, 2.55, 1.95)$, i.e.,

$$-2.23 \leq x \leq 2.27 \wedge -1.95 \leq y \leq 2.55$$

at loop head.

Solving the Equations

1. Decompose the system in **policies** (a.k.a., strategies);
2. Solve them using a linear programming solver.

Example

Hence $(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) = (2.27, 2.23, 2.55, 1.95)$, i.e.,

$$-2.23 \leq x \leq 2.27 \wedge -1.95 \leq y \leq 2.55$$

at loop head.

Remarks

- Above invariants is the least fixpoint.

Solving the Equations

1. Decompose the system in **policies** (a.k.a., strategies);
2. Solve them using a linear programming solver.

Example

Hence $(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) = (2.27, 2.23, 2.55, 1.95)$, i.e.,

$$-2.23 \leq x \leq 2.27 \wedge -1.95 \leq y \leq 2.55$$

at loop head.

Remarks

- Above invariants is the least fixpoint.
- Policy iterations are an efficient alternative to widening.
- However their use seems orthogonal to classic Kleene iterations.

Solving the Equations

1. Decompose the system in **policies** (a.k.a., strategies);
2. Solve them using a linear programming solver.

Example

Hence $(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) = (2.27, 2.23, 2.55, 1.95)$, i.e.,

$$-2.23 \leq x \leq 2.27 \wedge -1.95 \leq y \leq 2.55$$

at loop head.

Remarks

- Above invariants is the least fixpoint.
- Policy iterations are an efficient alternative to widening.
- However their use seems orthogonal to classic Kleene iterations.

$\Rightarrow$ Need to integrate policy iterations into a classic abstract domain.

Policy Iterations: Integration into a Classic Abstract Domain

Basic idea:

- A (symbolic) abstract domain "recording" the control flow graph.
- Product with a template domain.
- During widening:
  reduction of template domain thanks to policy iterations on:
  - **recorded control flow graph**;
  - and **appropriate templates**.

Policy Iterations: Integration into a Classic Abstract Domain

Basic idea:

- A (symbolic) abstract domain "recording" the control flow graph.
- Product with a template domain.
- During widening:
  reduction of template domain thanks to policy iterations on:
  - **recorded control flow graph**;
  - and **appropriate templates**.
    - Quadratic templates automatically obtained by solving Lyapunov equation.
    - Templates x^2 to get a bound on variable x.

Policy Iterations: Integration into a Classic Abstract Domain

Basic idea:

- A (symbolic) abstract domain "recording" the control flow graph.
- Product with a template domain.
- During widening:
  reduction of template domain thanks to policy iterations on:
  - **recorded control flow graph**;
  - and **appropriate templates**.
    - Quadratic templates automatically obtained by solving Lyapunov equation.
    - Templates x^2 to get a bound on variable x.

Result [RouxG-ATVA13]

A **classic abstract domain** able to
fully automatically infer quadratic invariants.

Summary

State of the Art

- Policy iterations [AdjéGG10, GawlitzaS10, GawlitzaSAGG12] promising technique for quadratic invariants, but:
  - Bad handling of linear constraints.
  - Initialization not automatic.
  - Require templates to be manually provided.
  - No formal soundness check for floating point issues.
  - No implementation in a static analyzer for classic code.

Contributions [RouxG-ATVA13]

Alleviate identified shortcomings of policy iterations:

- Handling linear constraints, initialization ([Roux-PhD13]).
- Automatic computation of templates.
- Soundness check.
- Integration of policy iterations in a classic abstract domain.

Implementation

- A new domain in our Lustre analyzer;
- 4kloc of OCaml and 0.5kloc of C code:
- implemented both min and max policies, using min by default;
- uses CSDP library for semidefinite programming;
- uses OCaml-GLPK for linear programming;
- soundness check of the result by Cholesky decompositions;
- available under a GPL license at `http://cavale.enseeiht.fr/policy2013/`.

Conclusion

On the code

x0 := 0; x1 := 0; x2 := 0;
while $-1 \leq 0$ **do**
  in := **?(**-1, 1**)**;
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od

providing only the list of variables x_0, x_1, x_2, *in*
we want to apply our domain on, our tool computes

$$6.2547x_0^2 + 12.1868x_1^2 + 3.8775x_2^2 - 10.61x_0x_1 - 2.4306x_0x_2 + 2.4182x_1x_2 \leq 1.0029$$
$$\wedge x_0^2 \leq 0.1795 \wedge x_1^2 \leq 0.1136 \wedge x_2^2 \leq 0.2757$$

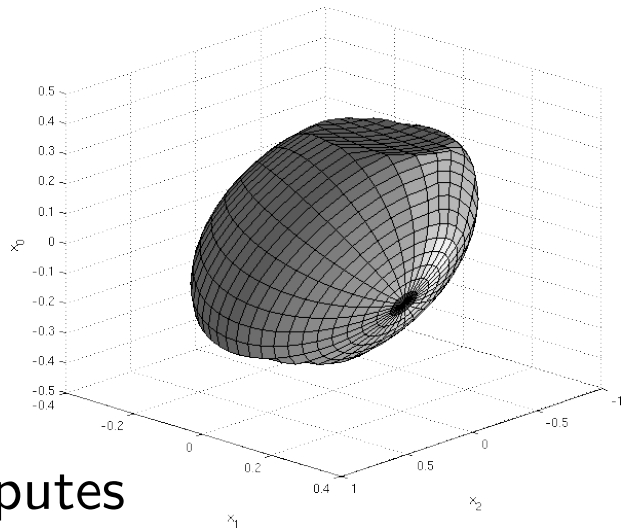
then

$$|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251.$$

Conclusion

On the code

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
   in := ?(−1, 1);
   x0' := x0; x1' := x1; x2' := x2;
   x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
   x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
   x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```



providing only the list of variables x_0, x_1, x_2, *in* we want to apply our domain on, our tool computes

$$6.2547x_0^2 + 12.1868x_1^2 + 3.8775x_2^2 - 10.61x_0x_1 - 2.4306x_0x_2 + 2.4182x_1x_2 \leq 1.0029$$
$$\wedge x_0^2 \leq 0.1795 \wedge x_1^2 \leq 0.1136 \wedge x_2^2 \leq 0.2757$$

then

$$|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251.$$

1 Quadratic Invariants

2 Policy Iterations

3 Open Directions

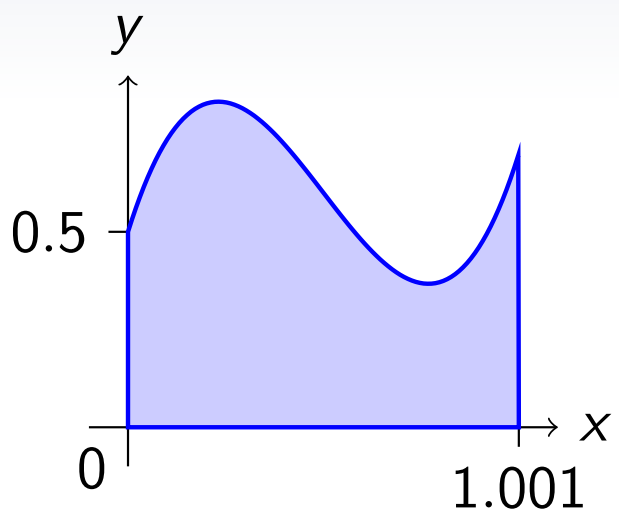
Other Properties

- Invariant generation:
  - Common Lyapunov functions.
  - Piecewise linear quadratic Lyapunov functions [RantzerJ00] (joint work with A. Adjé, ongoing work).
- Closed loop system:
  - Discretization of the plant.
  - Stability (joint work with R. Jobredeaux, submitted).
    - Generation of templates in presence of saturations.
    - Elimination of redundant variables.
  - Robustness (generalization of phase and gain margins).
  - Performance (overshoot, convergence speed).

Polynomial Invariants on Polynomial Programs

Classical template abstract domain based on **Bernstein polynomials** global optimization [RouxG-NSV13].

```
x := 0; y := ?(0, 0.5);
while x ≤ 1 do
    y := y + 0.001 × (18x² − 18x + 3);
    x := x + 0.001;
    if y ≤ 0 then y := 0 else y := y fi
od
```

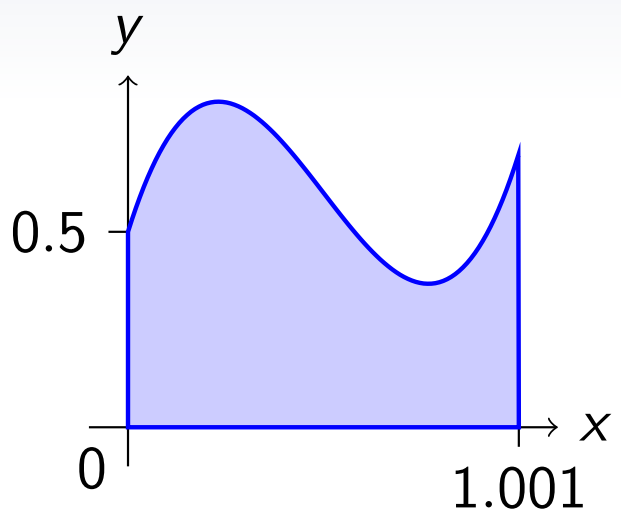


$$
\begin{array}{rcl}
y - 6x^3 + 9x^2 - 3.2x & \leq & 0.5 \\
x & \leq & 1.001 \\
-x & \leq & 0 \\
y & \leq & 0.833 \\
-y & \leq & 0
\end{array}
$$

Polynomial Invariants on Polynomial Programs

Classical template abstract domain based on **Bernstein polynomials** global optimization [RouxG-NSV13].

```
x := 0; y := ?(0, 0.5);
while x ≤ 1 do
    y := y + 0.001 × (18x² − 18x + 3);
    x := x + 0.001;
    if y ≤ 0 then y := 0 else y := y fi
od
```



$$
\begin{array}{rcl}
y - 6x^3 + 9x^2 - 3.2x & \leq & 0.5 \\
x & \leq & 1.001 \\
-x & \leq & 0 \\
y & \leq & 0.833 \\
-y & \leq & 0
\end{array}
$$

Ideas

- Automatic generation of templates (based on SOS methods [KordaHJ13]).
- Extension to non algebraic functions (Taylor or Poisson expansion, beginning collaboration with G. Morin and X. Thirioux).