Integrating Policy Iterations in Abstract Interpreters

Pierre Roux [1,2] **Pierre-Loïc Garoche** [1]

October 17, 2013

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

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

Example

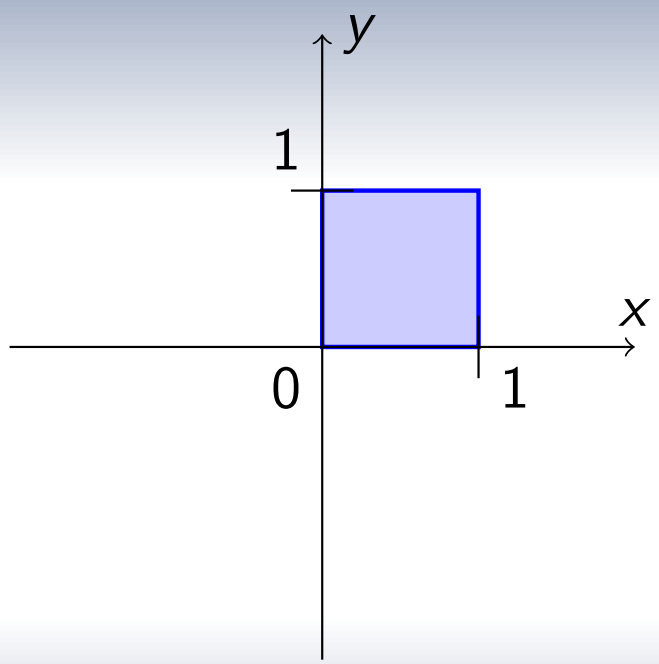
Example

$x := ?(0, 1)$; $y := ?(0, 1)$;
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

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
```

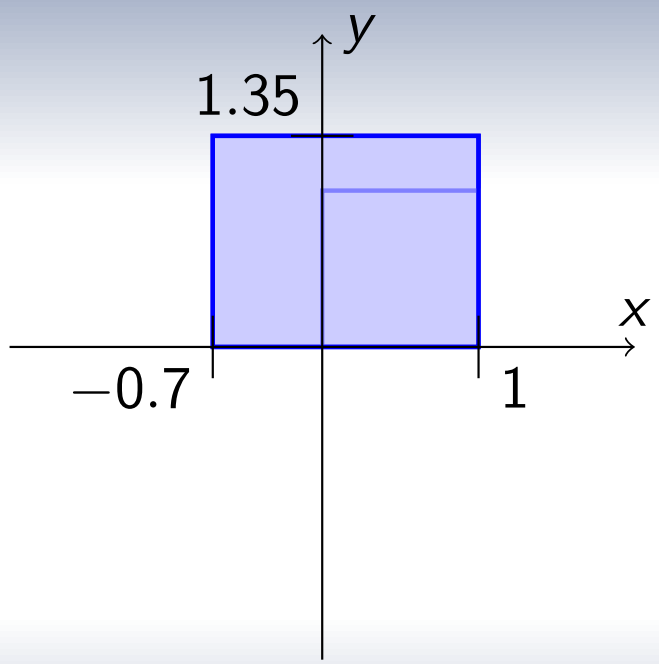


before entering the loop

Kleene Iterations

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
```

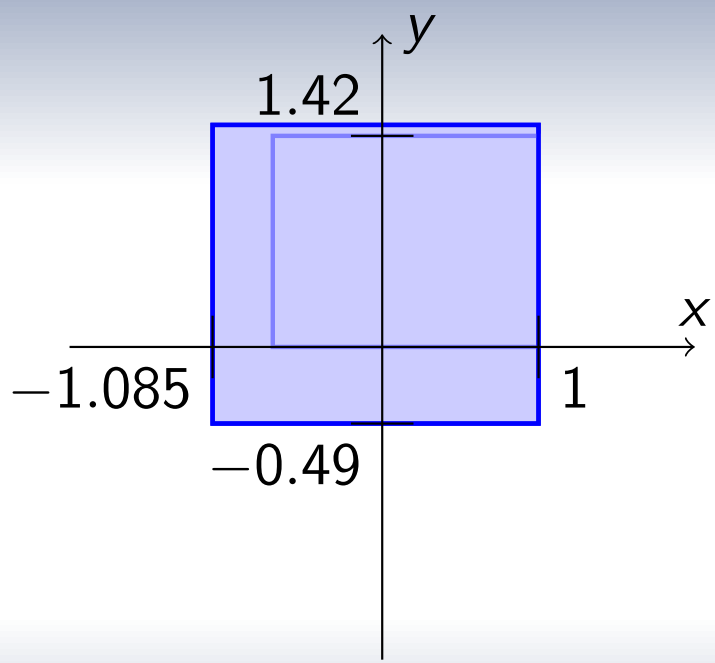


after a first iteration

Kleene Iterations

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
```

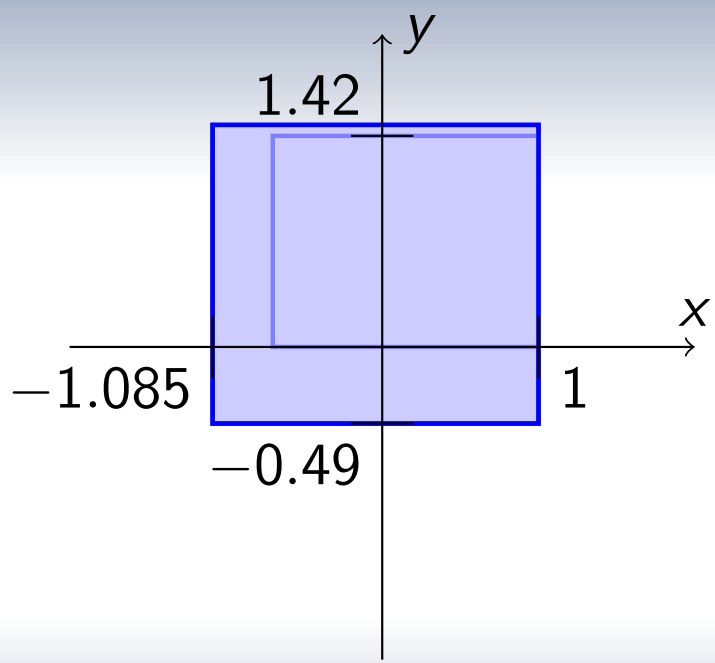


after a second iteration

Kleene Iterations

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

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

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

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

Basic Idea

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

Example (Simplest Widening)

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

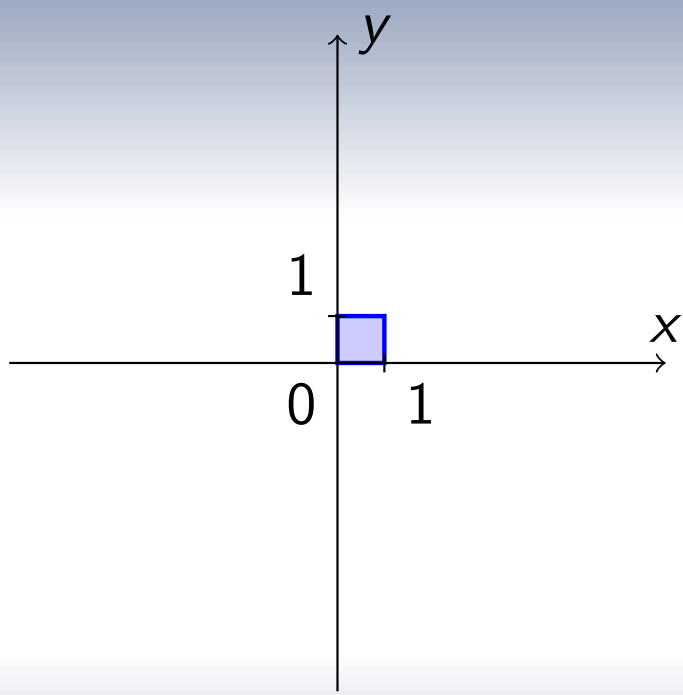
Leads to $x \in (\infty, +\infty) \wedge y \in (-\infty, +\infty)$: useless result.

Example (Widening with Thresholds)

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

Widening with Thresholds

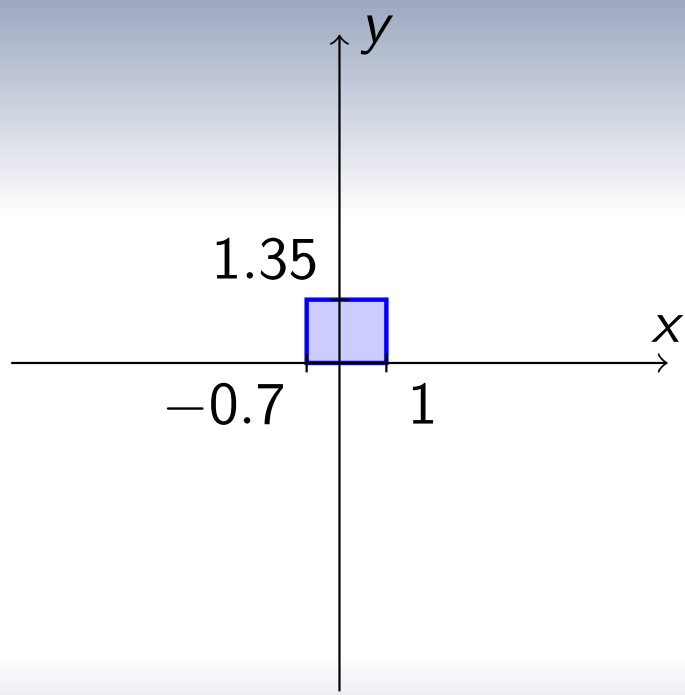
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



before entering the loop

Widening with Thresholds

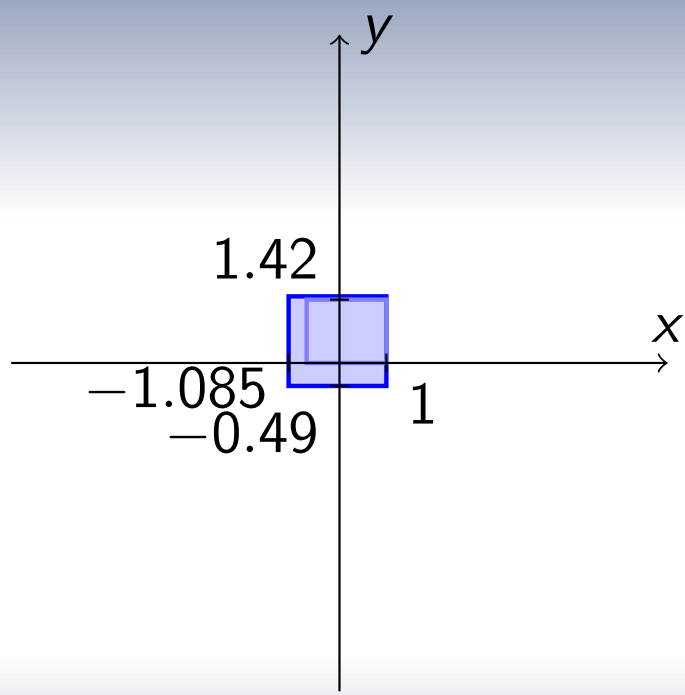
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after a first iteration

Widening with Thresholds

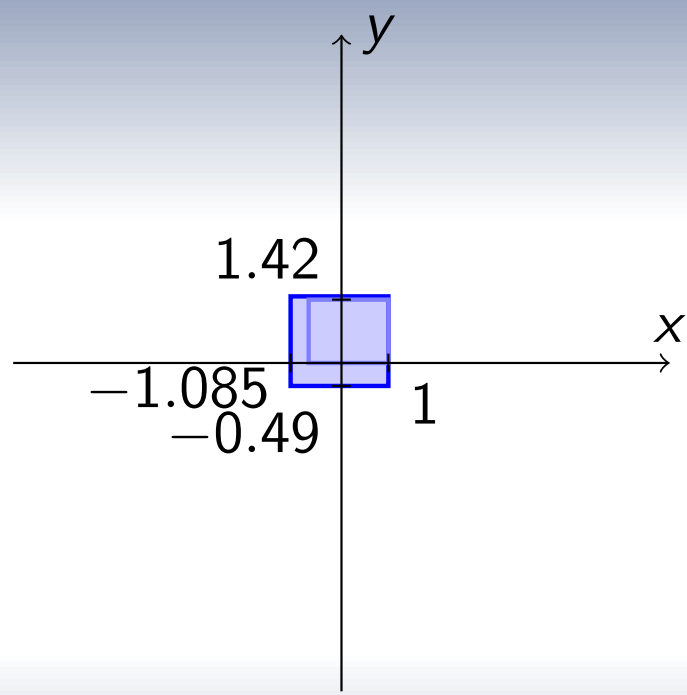
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after a second iteration

Widening with Thresholds

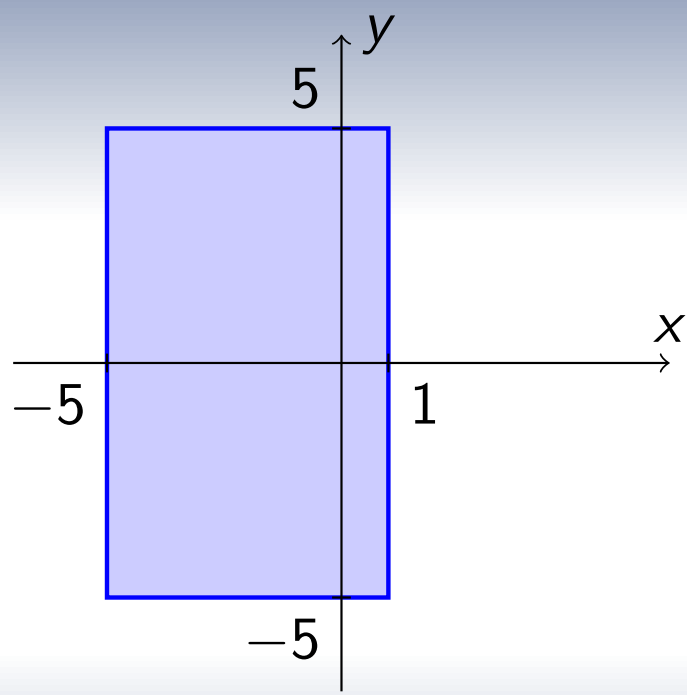
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



not stable → widening

Widening with Thresholds

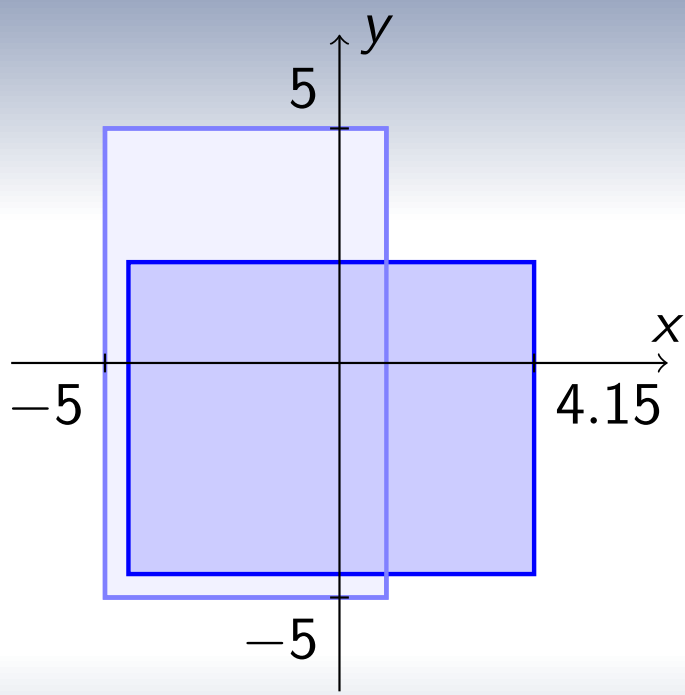
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after widening

Widening with Thresholds

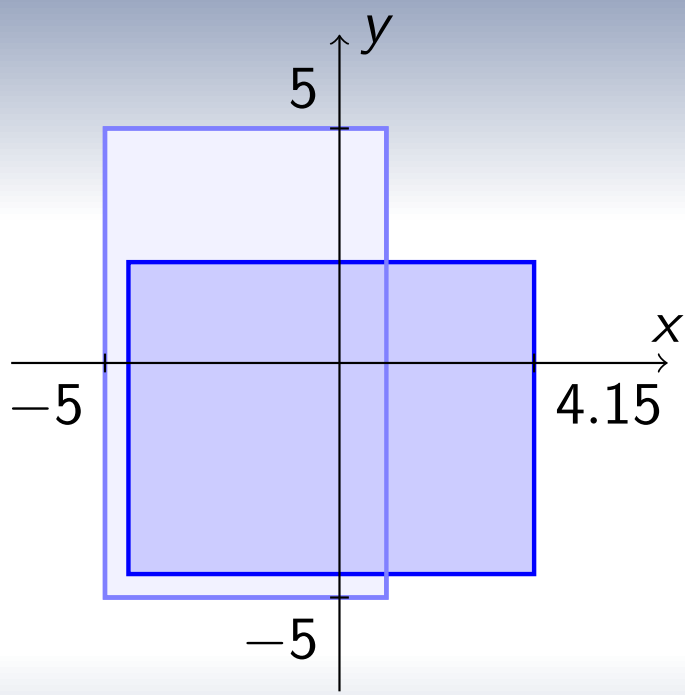
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after another iteration

Widening with Thresholds

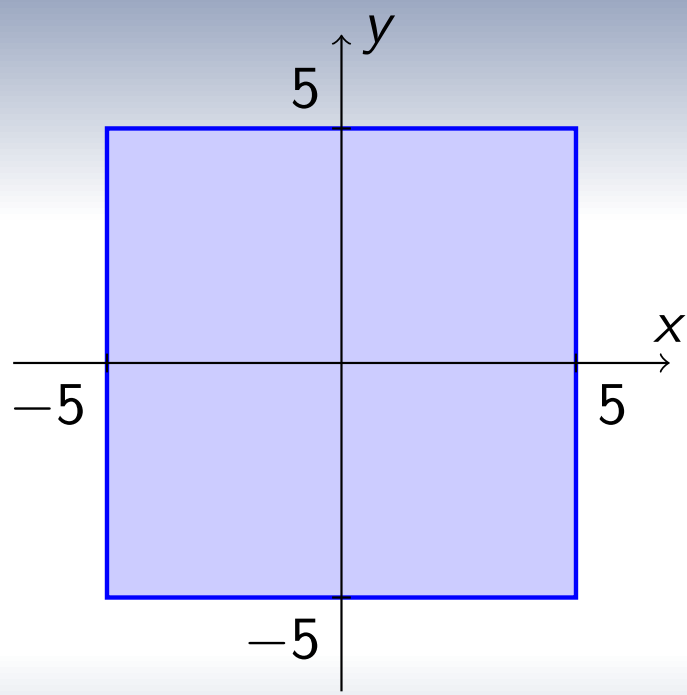
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



not stable → widening

Widening with Thresholds

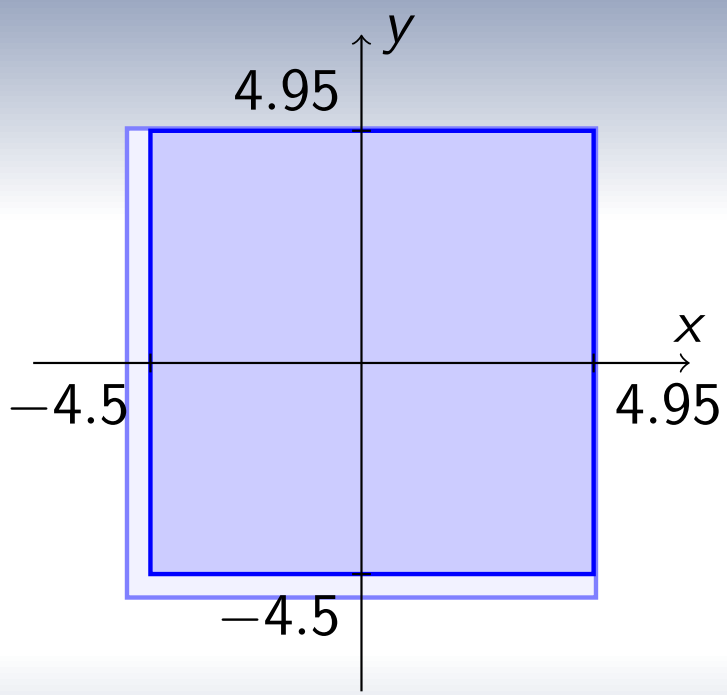
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after widening

Widening with Thresholds

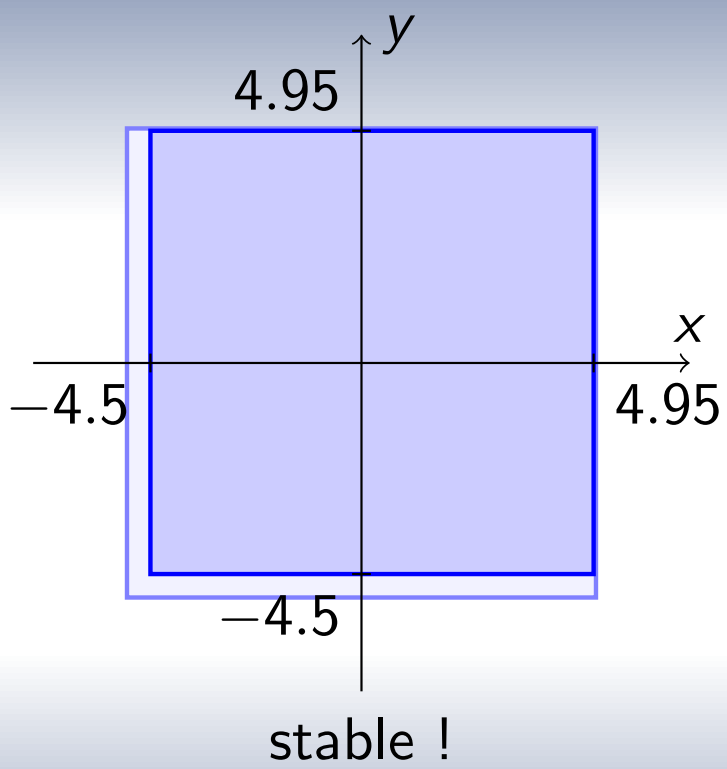
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after another iteration

Widening with Thresholds

```
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
```



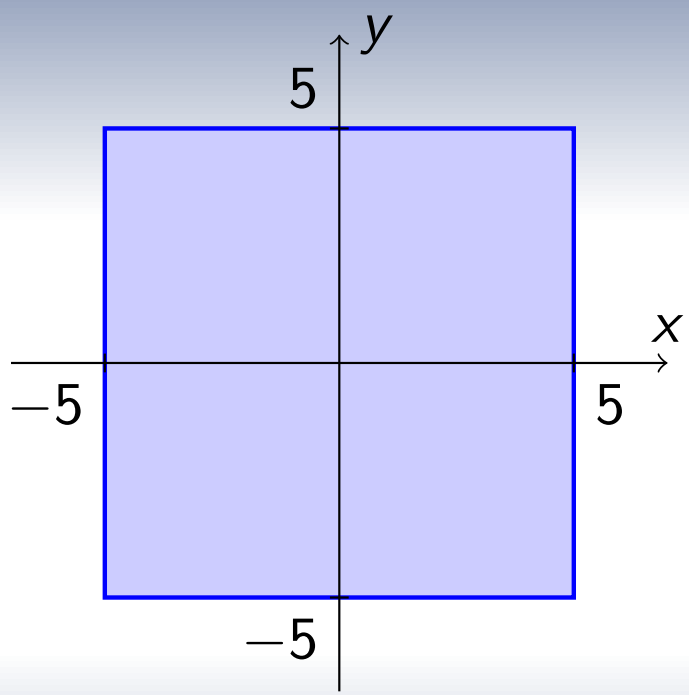
Descending Iterations with Narrowing

Perform a few descending iterations:

- allows to retrieve precision lost in widening;
- cannot always regain everything.

Descending Iterations with Narrowing

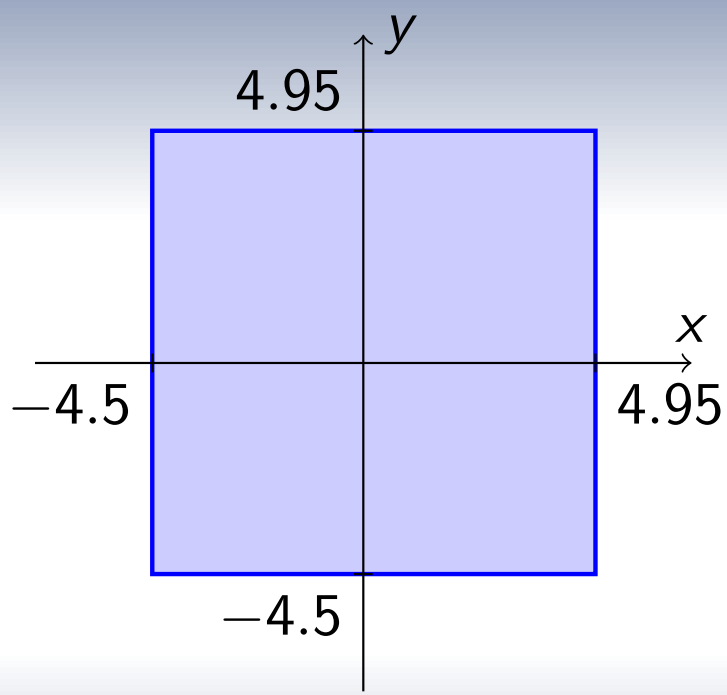
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



fixpoint reached with widening

Descending Iterations with Narrowing

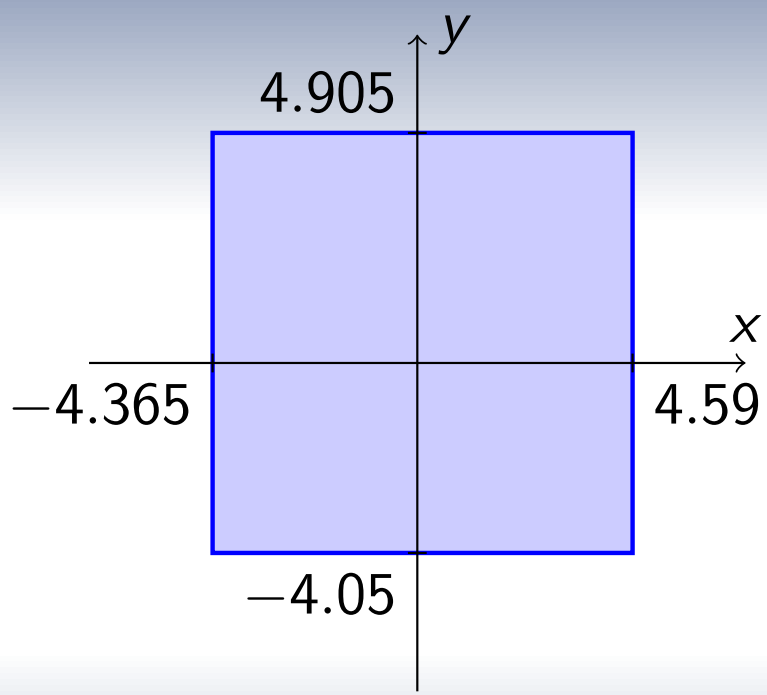
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after a descending iteration

Descending Iterations with Narrowing

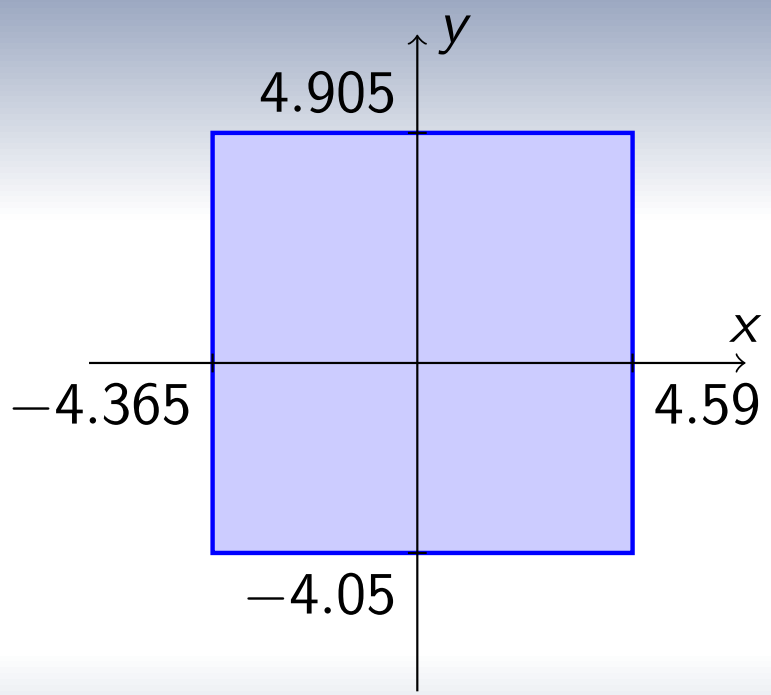
```
x := ?(0, 1); y := ?(0, 1);
while −1 ≤ 0 do
   in := ?(0, 1);
   if 0.9 − in ≤ 0 then
      x := 10×in − 9;
      y := 10×in − 9
   else
      t := x;
      x := 0.2×t − 0.7×y + 0.5×in;
      y := 0.7×t + 0.2×y + 0.5×in
   fi
od
```



after a second iteration

Descending Iterations with Narrowing

```
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 converging, we stop here

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

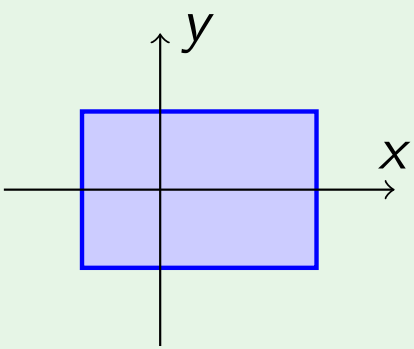
Template Domains

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

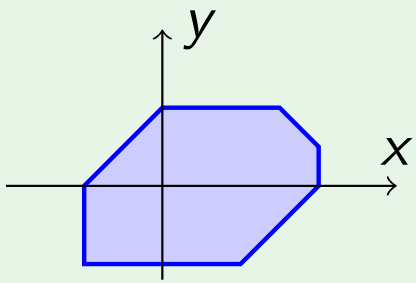
$(2, 1, 1, 1)$



intervals

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

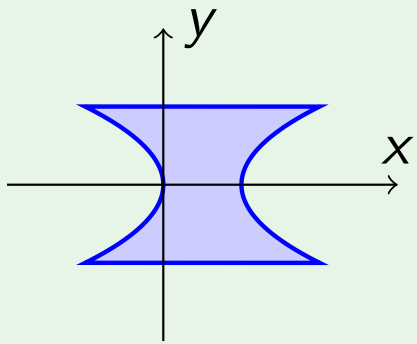
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

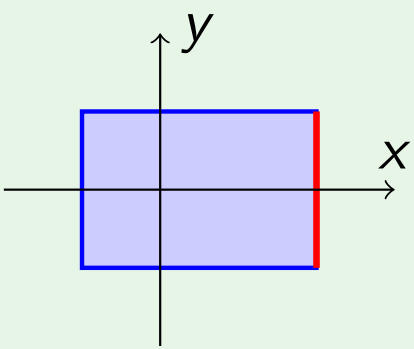
Template Domains

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

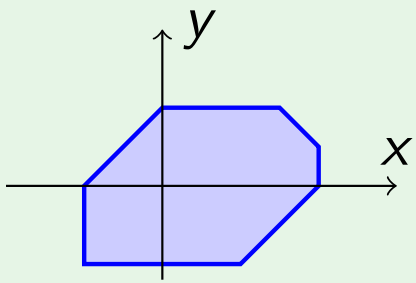
$(2, 1, 1, 1)$



intervals

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

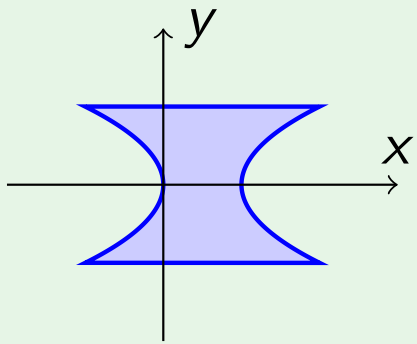
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

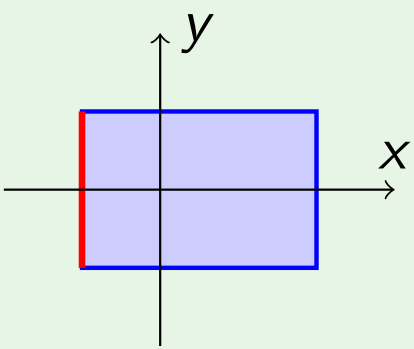
Template Domains

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

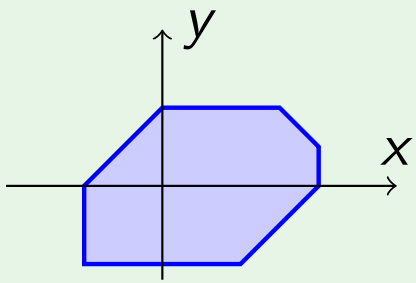
$(2, 1, 1, 1)$



intervals

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

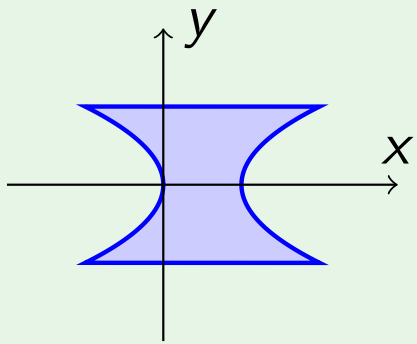
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

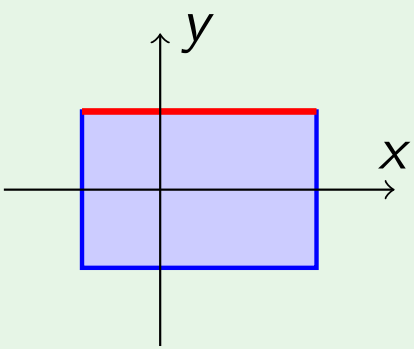
Template Domains

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

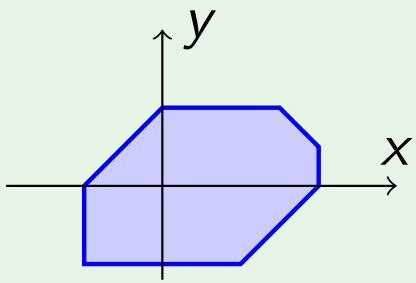
$(2, 1, 1, 1)$



intervals

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

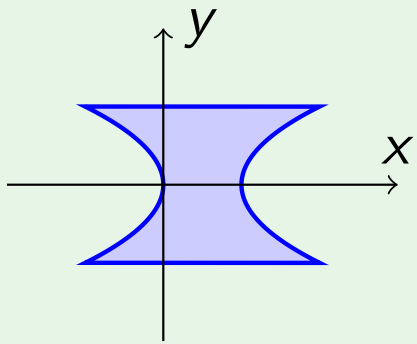
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

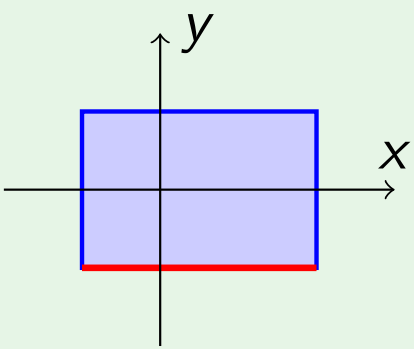
Template Domains

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

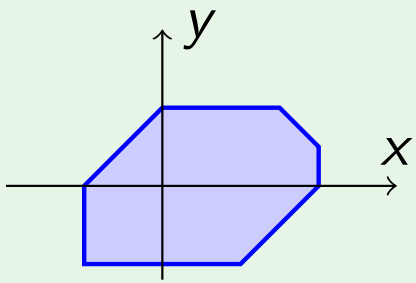
$(2, 1, 1, 1)$



intervals

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

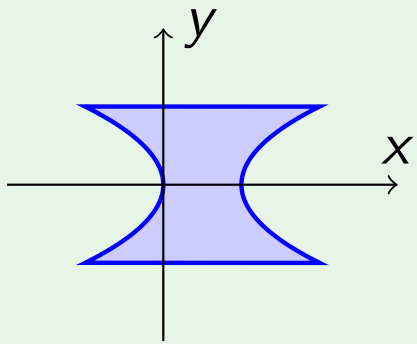
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

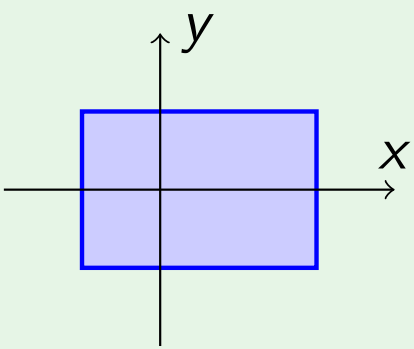
Template Domains

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

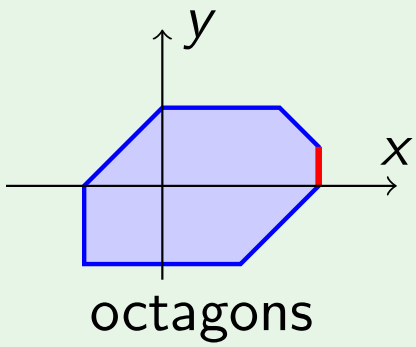
$(2, 1, 1, 1)$



intervals

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

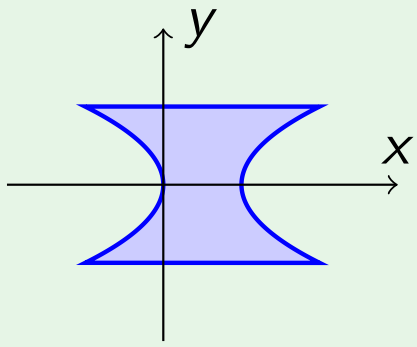
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

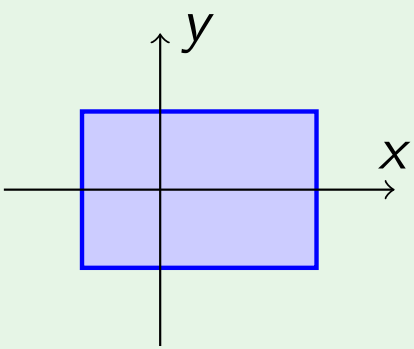
Template Domains

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

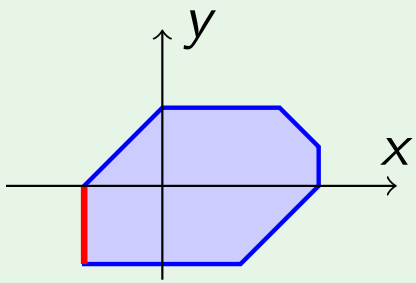
$(2, 1, 1, 1)$



intervals

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

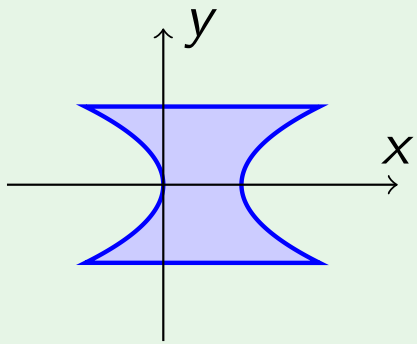
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

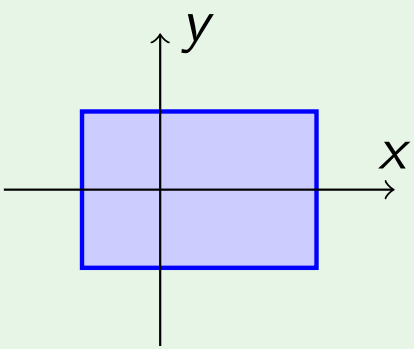
Template Domains

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

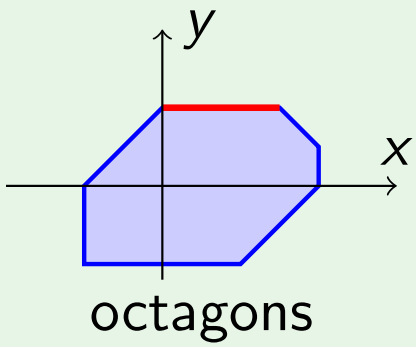
$(2, 1, 1, 1)$



intervals

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

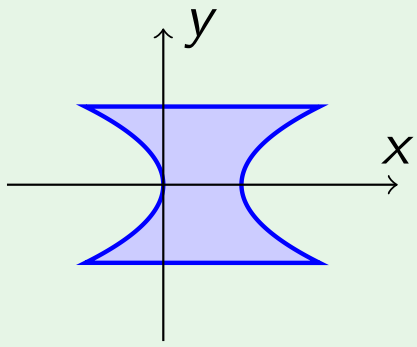
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

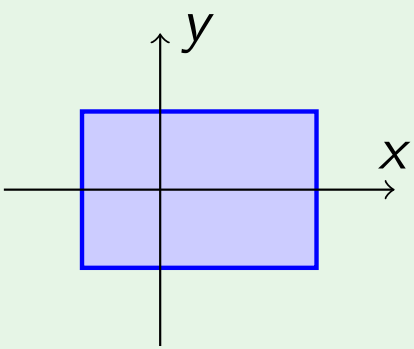
Template Domains

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

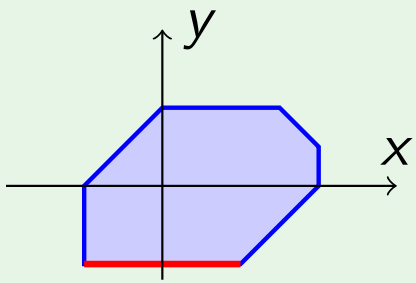
$(2, 1, 1, 1)$



intervals

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

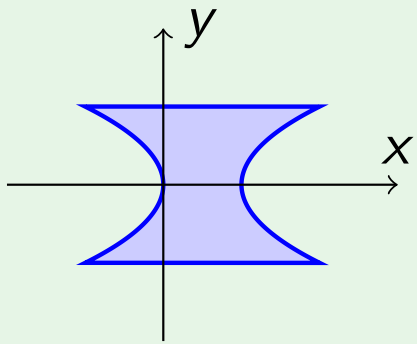
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

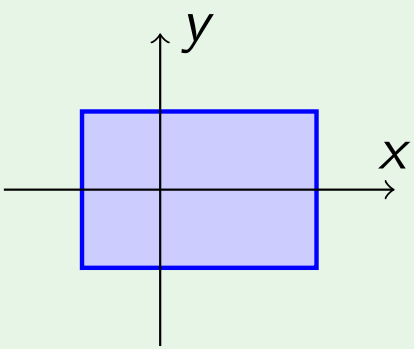
Template Domains

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

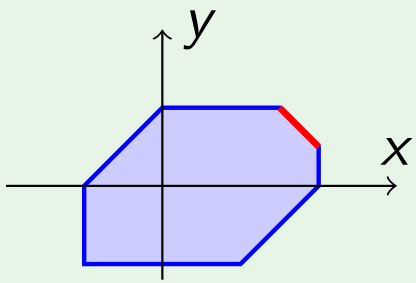
$(2, 1, 1, 1)$



intervals

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

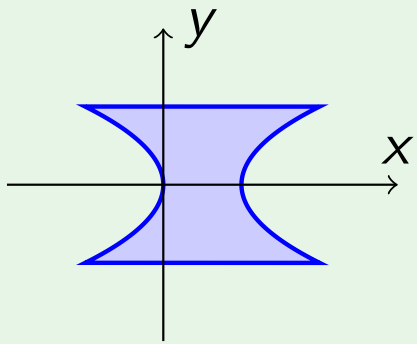
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

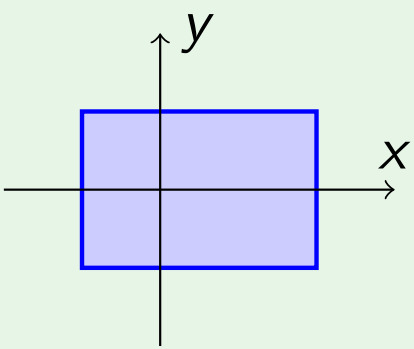
Template Domains

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

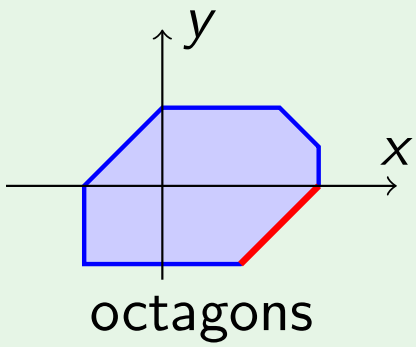
$(2, 1, 1, 1)$



intervals

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

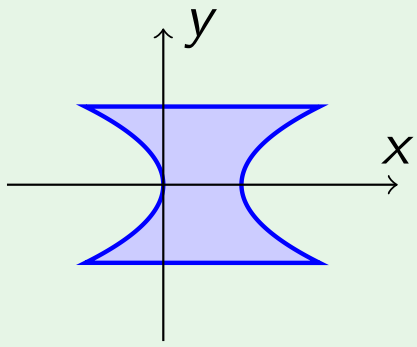
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

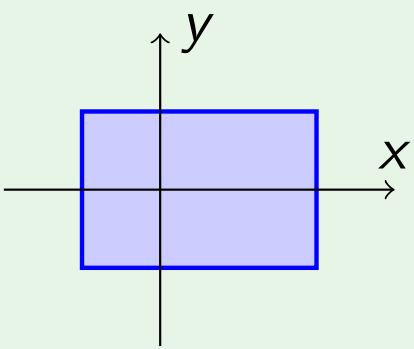
Template Domains

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

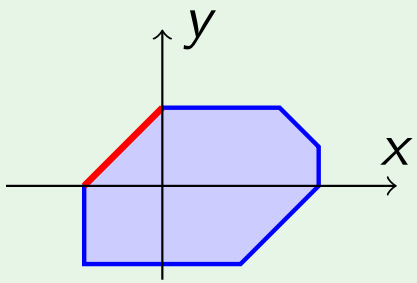
$(2, 1, 1, 1)$



intervals

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

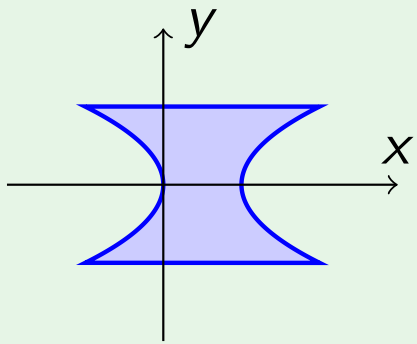
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

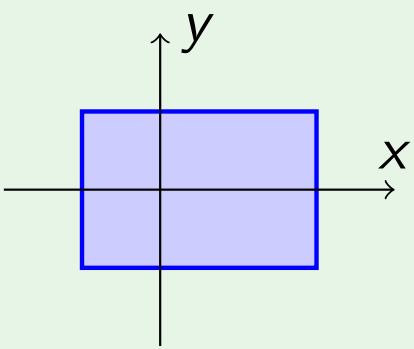
Template Domains

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

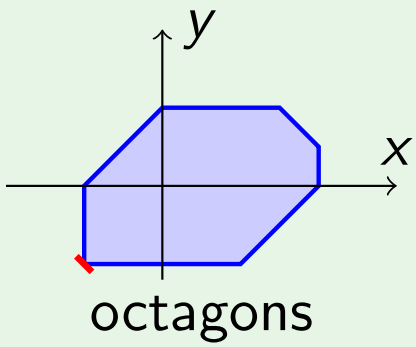
$(2, 1, 1, 1)$



intervals

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

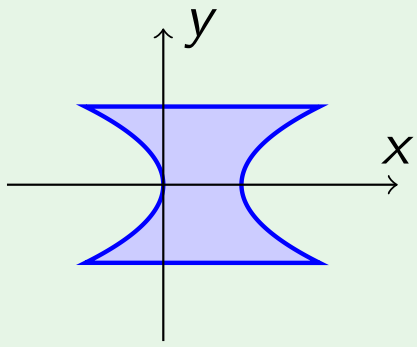
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

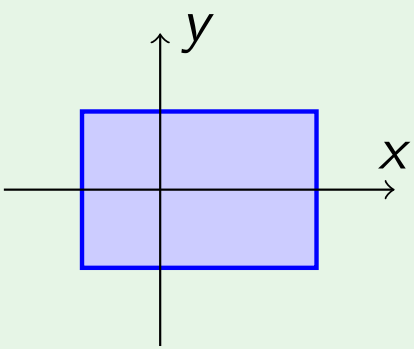
Template Domains

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

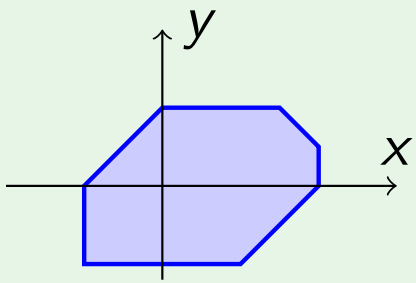
$(2, 1, 1, 1)$



intervals

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

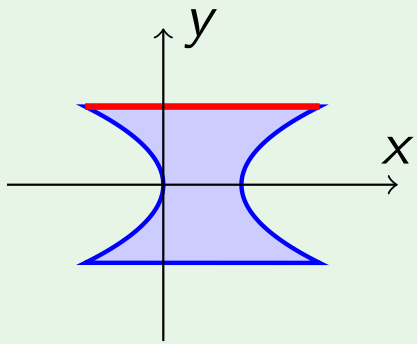
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

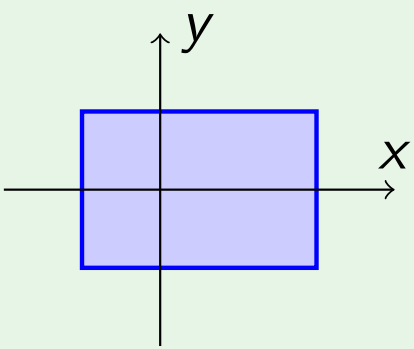
Template Domains

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

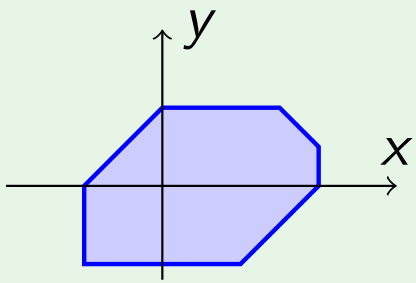
$(2, 1, 1, 1)$



intervals

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

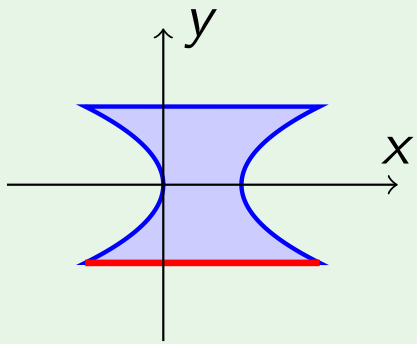
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

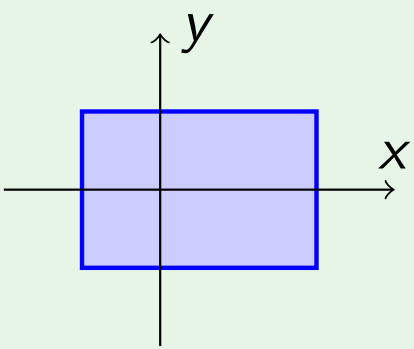
Template Domains

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

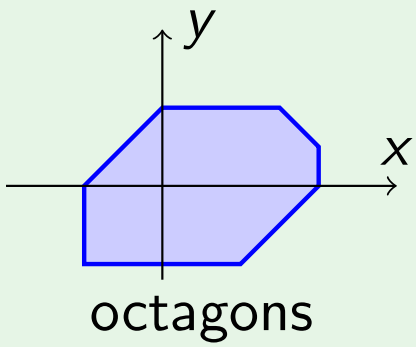
$(2, 1, 1, 1)$



intervals

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

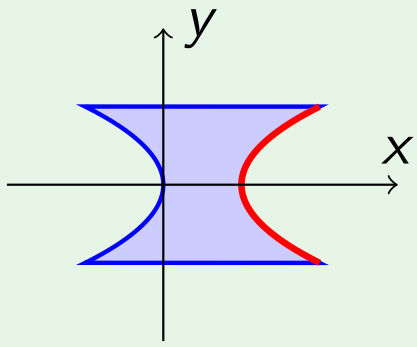
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

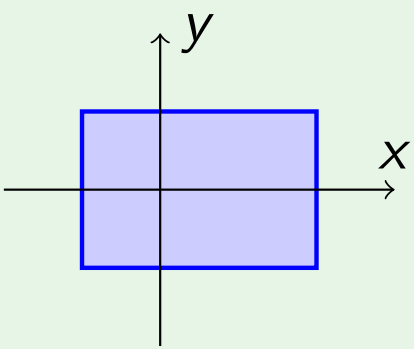
Template Domains

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

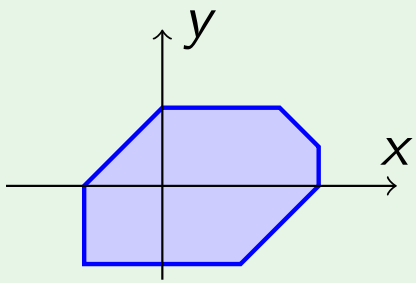
$(2, 1, 1, 1)$



intervals

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

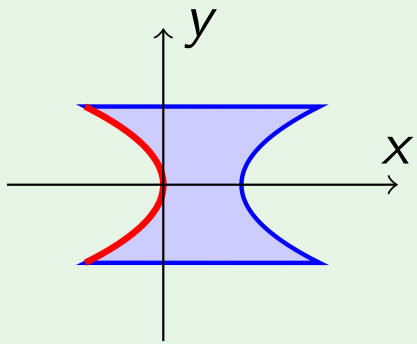
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



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

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

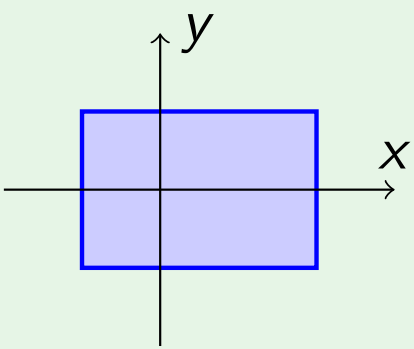
Template Domains

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

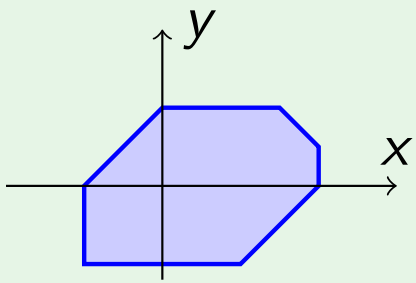
$(2, 1, 1, 1)$



intervals

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

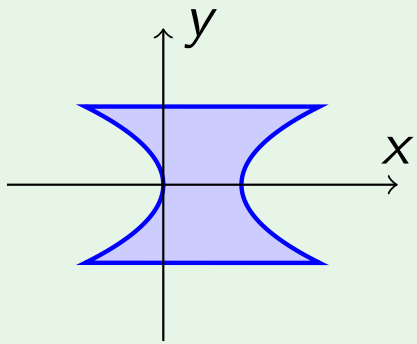
$(2, 1, 1, 1, 2.5, 2, 1, 2)$



octagons

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

$(1, 1, 1, 0)$

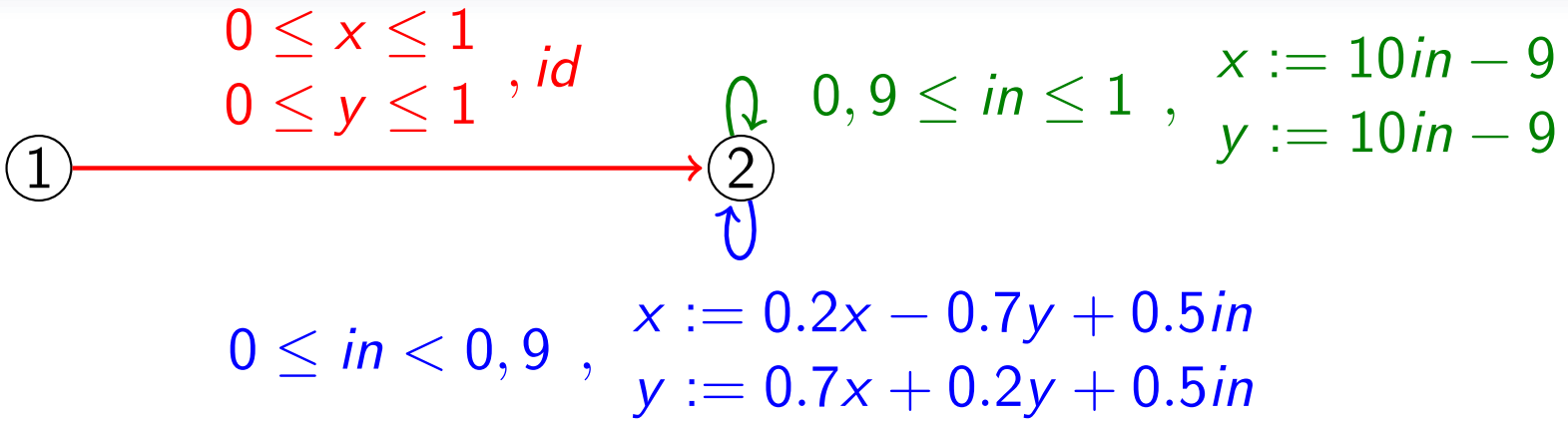


quadratic

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

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

Solving the Equations

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

Solving the Equations

1. Decompose the system in **policies** (aka 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.

Policy Iterations vs Kleene Iterations

- policy iterations are an efficient alternative to widening

Policy Iterations vs Kleene Iterations

- policy iterations are an efficient alternative to widening
- however their use seems orthogonal to traditional Kleene iterations

Policy Iterations vs Kleene Iterations

- policy iterations are an efficient alternative to widening
- however their use seems orthogonal to traditional Kleene iterations
- this prevents an easy use in existing static analyzers
  and collaboration with existing domains through reduced products

Policy Iterations vs Kleene Iterations

- policy iterations are an efficient alternative to widening
- however their use seems orthogonal to traditional Kleene iterations
- this prevents an easy use in existing static analyzers
  and collaboration with existing domains through reduced products

$\Rightarrow$ we want to integrate policy iterations
into a traditional abstract domain

Policy Iterations: Integration into a Traditional Abstract Domain

- An abstract domain "recording" the control flow graph (and potential info from other domains).

Policy Iterations: Integration into a Traditional Abstract Domain

- An abstract domain "recording" the control flow graph (and potential info from other domains).
- Product with a template domain.

Policy Iterations: Integration into a Traditional Abstract Domain

- An abstract domain "recording" the control flow graph (and potential info from other domains).
- Product with a template domain.
- Use policy iterations on control flow graph and templates as a precise widening
  (result can be then exported to other domains).

Example of Control Command Program

Example

x0 := 0; x1 := 0; x2 := 0;
while $-1 \leq 0$ **do**
  in := ?(-1, 1);
  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
  x0 := x0'; x1 := x1'; x2 := x2';
od

Example of Control Command Program

Example

x0 := 0; x1 := 0; x2 := 0;
while $-1 \leq 0$ **do**
  in := ?(-1, 1);
  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
  x0 := x0'; x1 := x1'; x2 := x2';
od

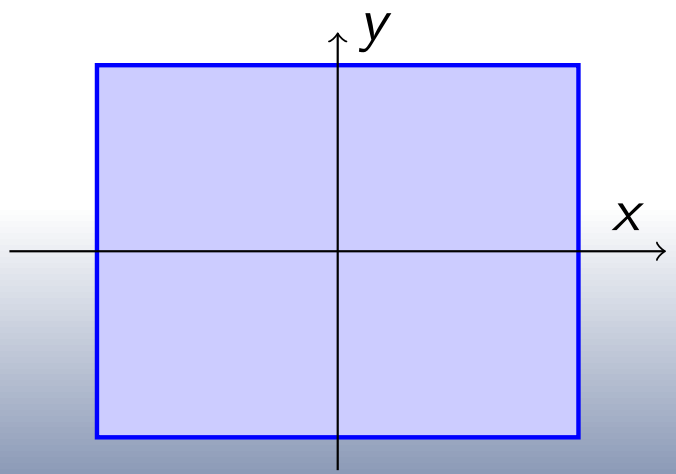
No invariant in intervals domain!

Quadratic invariants

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

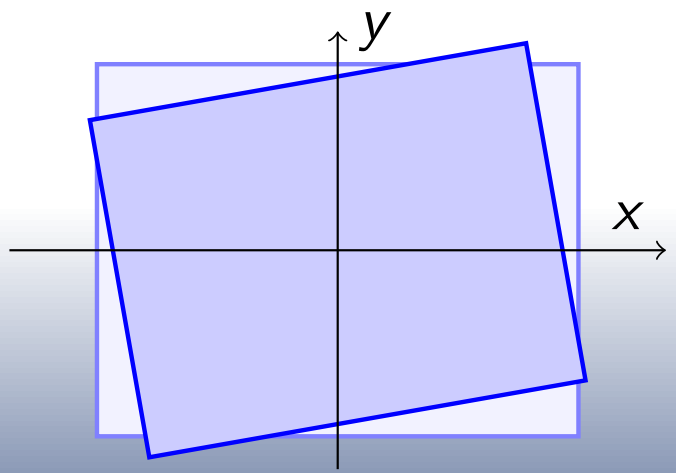
Quadratic invariants

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



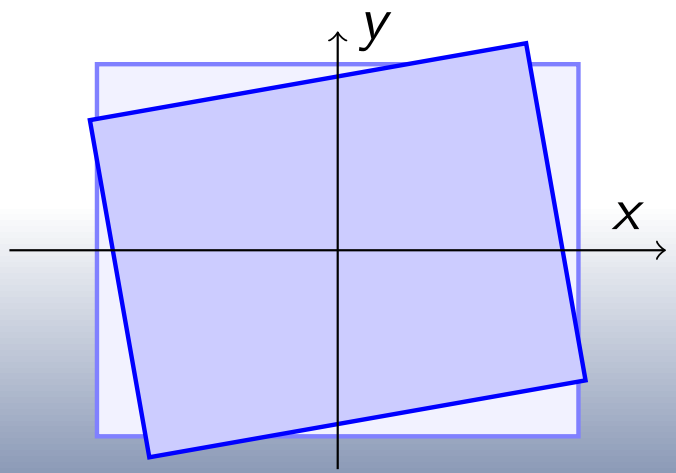
Quadratic invariants

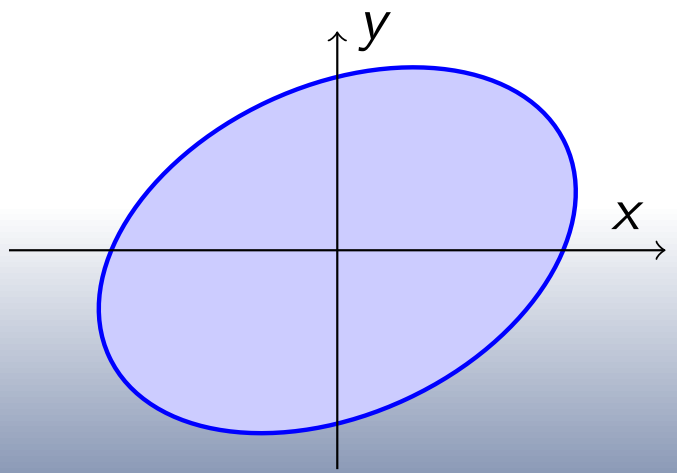
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



Quadratic invariants

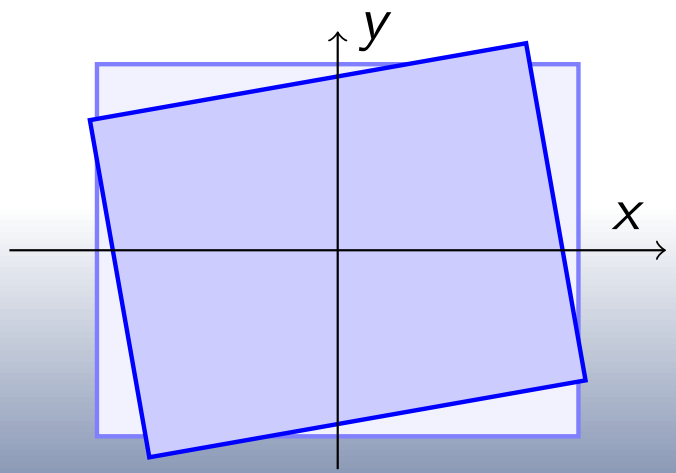
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that **quadratic invariants** are a good fit for linear systems.

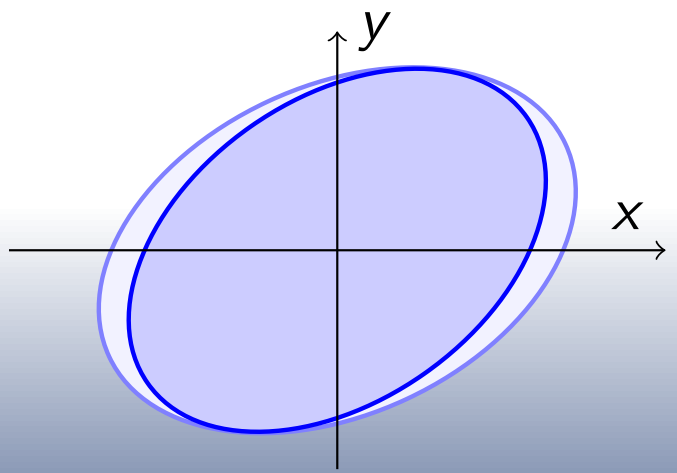




Quadratic invariants

- **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 Templates

- Control theorist generate suitable quadratic templates by numerically solving a so called **Lyapunov equation**.
- Templates x_i^2 can be added for each variable x_i to get bounds on these variables.

Quadratic Templates

- Control theorist generate suitable quadratic templates by numerically solving a so called **Lyapunov equation**.
- Templates x_i^2 can be added for each variable x_i to get bounds on these variables.

Example

The set of templates $\mathcal{T} = (t_1, t_2, t_3, t_4)$ can be used with

$$t_1 := 6.25x_0^2 + 12.19x_1^2 + 3.88x_2^2 - 10.61x_0x_1 - 2.43x_1x_2 + 2.42x_1x_2$$
$$t_2 := x_0^2$$
$$t_3 := x_1^2$$
$$t_4 := x_2^2$$

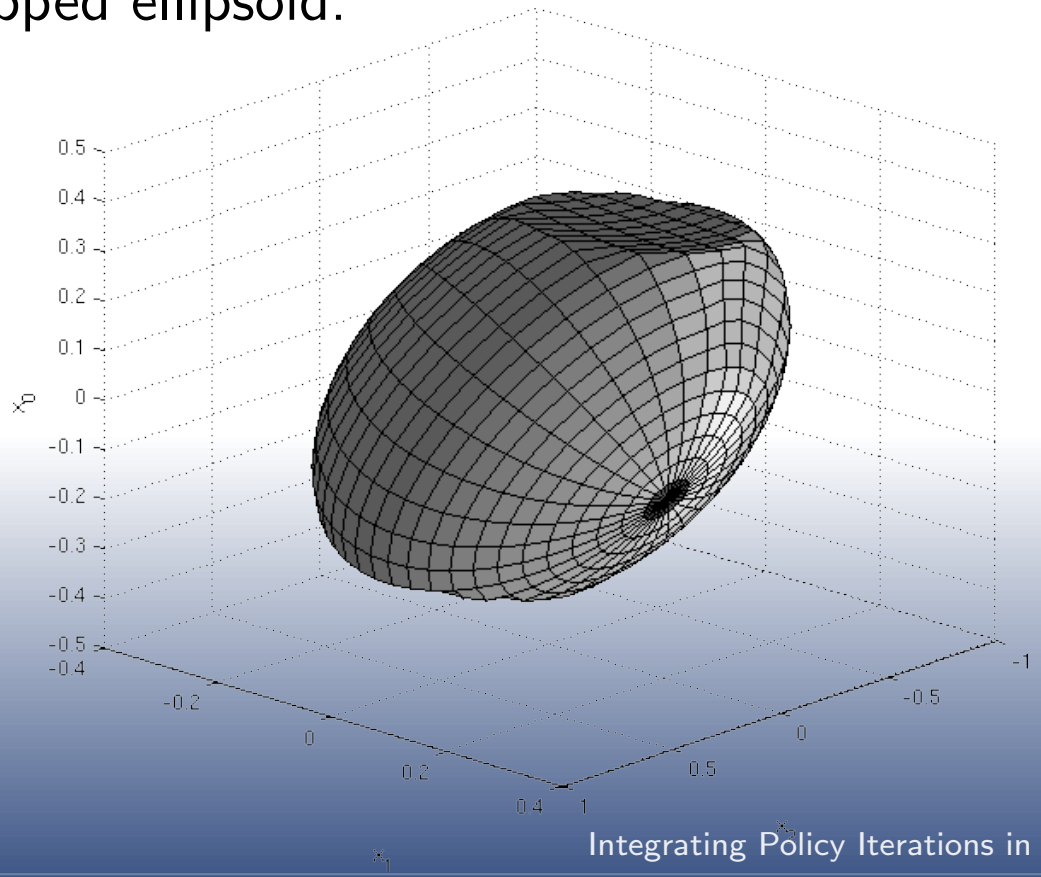
Policy Iterations: Result

Policy iterations compute at loop head the invariant
$(1.0029, 0.1795, 0.1136, 0.2757) \in \mathcal{T}$,
meaning: $t_1 \leq 1.0029 \wedge x_0^2 \leq 0.1795 \wedge x_1^2 \leq 0.1136 \wedge x_2^2 \leq 0.2757$
i.e.: $t_1 \leq 1.0029 \wedge |x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

Policy Iterations: Result

Policy iterations compute at loop head the invariant
$(1.0029, 0.1795, 0.1136, 0.2757) \in \mathcal{T}$,
meaning: $t_1 \leq 1.0029 \wedge x_0^2 \leq 0.1795 \wedge x_1^2 \leq 0.1136 \wedge x_2^2 \leq 0.2757$
i.e.: $t_1 \leq 1.0029 \wedge |x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$.

This is a cropped ellipsoid:



Floating Point Issues

- Rounding errors in the analyzed program: todo.
- Rounding errors in the analyzer:
  - Checking the soundness of the result basically amounts to **checking positive definiteness** of a matrix.
  - This is done by carefully bounding the rounding errors in a **Cholesky decomposition**.
  - Hence an efficient soundness check (in $\mathrm{O}\left(n^3\right)$ floating point operations for an $n \times n$ matrix).

Implementation

- A new domain in our Lustre analyzer;
- 4kloc of OCaml and 0.5kloc of C code;
- uses CSDP library for semidefinite programming;
- uses OCaml-GLPK for linear programming;
- available under a GPL license at `http://cavale.enseeiht.fr/policy2013/`.

Questions

Thank you for your attention!

?

Stability of Linear Systems

- We assume a system of the form

$$x_{k+1} = Ax_k + Bu_k.$$

- And bounds on inputs u_k.
- We want to bound x_k.

Lyapunov Stability

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

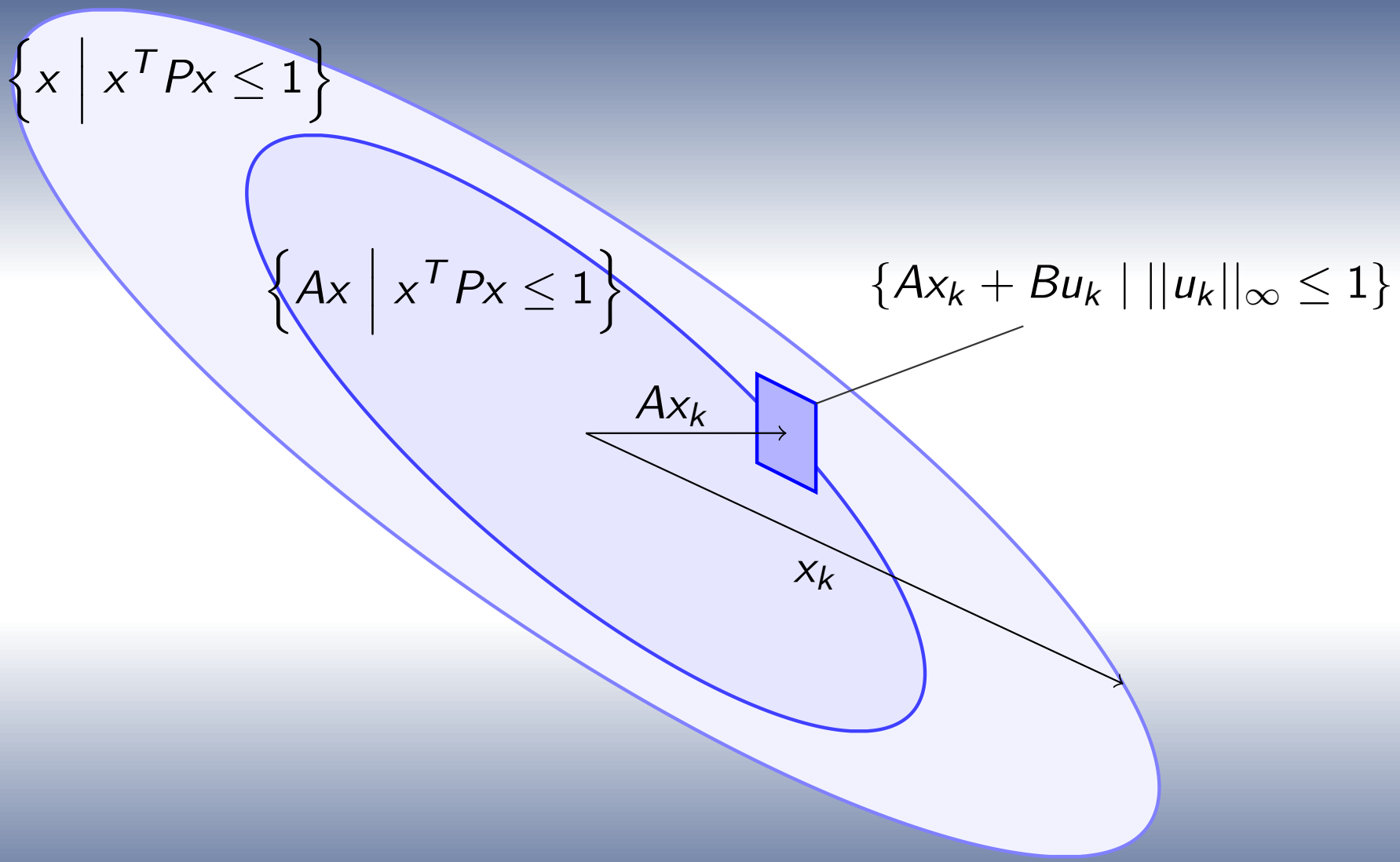
Invariant ellipsoid

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



Method

Overall Method

1. First determine the **shape** of the ellipsoid by choosing a matrix P;
2. then find the smallest possible **ratio** λ such that $x^T P x \leq \lambda$ is an invariant.

Tools

Definition (Semidefinite Programming)

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 "$P \succeq 0$" means $x^T P x \geq 0$ for all vector x.

Tools

Definition (Semidefinite Programming)

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 "$P \succeq 0$" means $x^T P x \geq 0$ for all vector x.

- "Efficient" solvers exist;

Tools

Definition (Semidefinite Programming)

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 "$P \succeq 0$" means $x^T P x \geq 0$ for all vector x.

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

Tools

Definition (Semidefinite Programming)

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 "$P \succeq 0$" means $x^T P x \geq 0$ for all vector x.

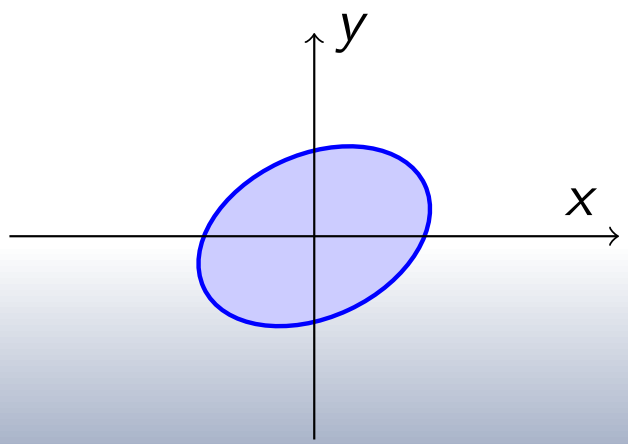
- "Efficient" solvers exist;
- Unknown matrices: Y can be decomposed as $\sum_{i,j} y_{i,j} E^{i,j}$;
- Multiple constraints: $A \succeq 0 \wedge B \succeq 0$ is equivalent to $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \succeq 0$.

Shape of the Ellipsoid

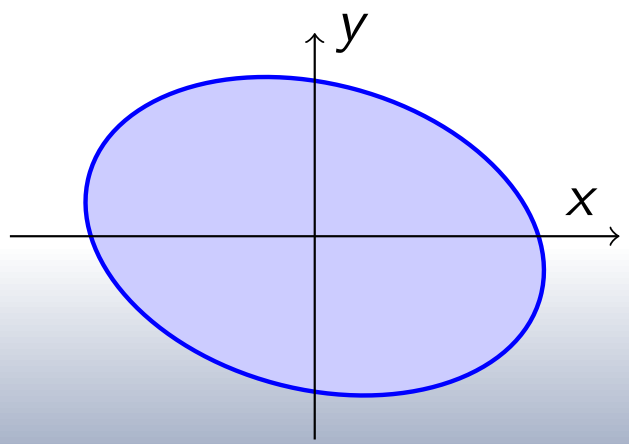
- We look for a positive definite matrix P such that $P - A^T P A \succ 0$.

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

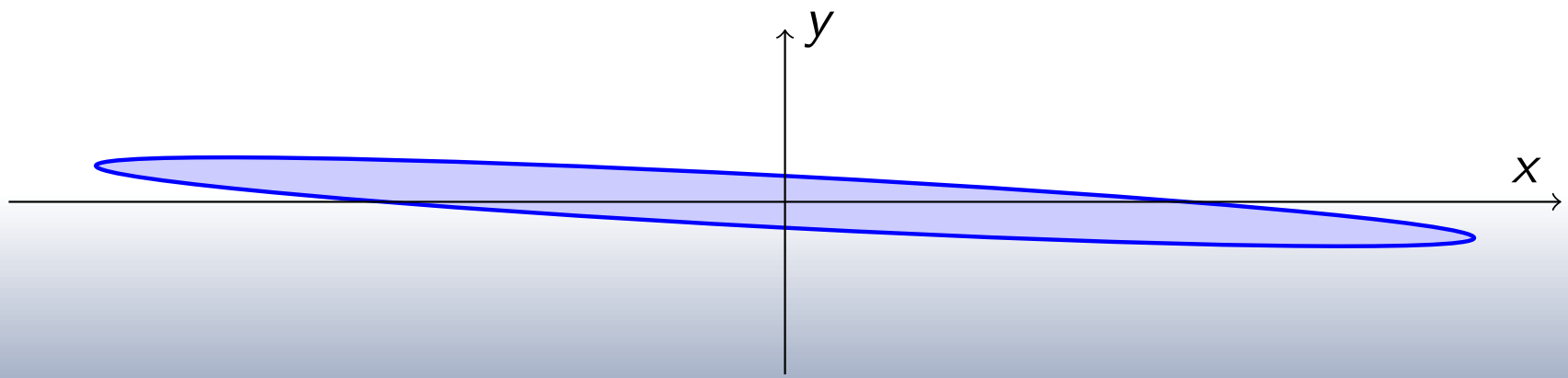


Heuristic 1: Minimizing Condition Number

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.

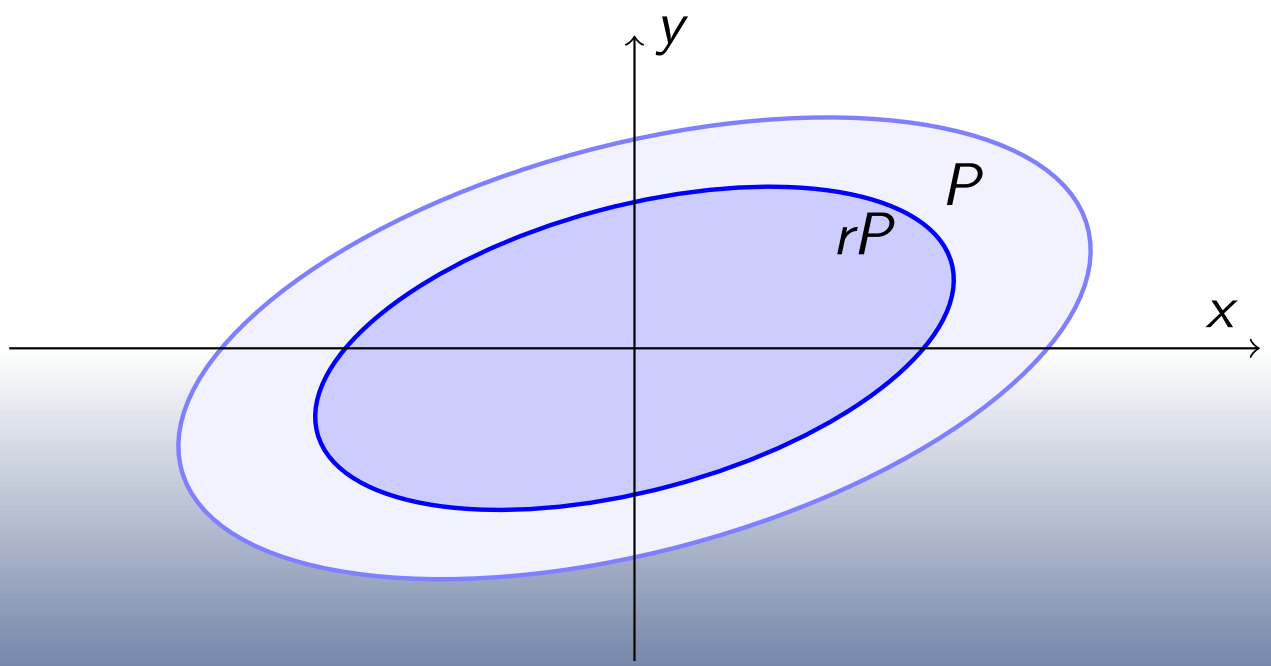


Heuristic 2: Preserving the Shape

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.

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

$$\begin{pmatrix} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & 1 - e_i^T B^T PBe_i \end{pmatrix} - \tau_i \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

for all the vertices e_i of the hypercube of dimension p.

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

$$\begin{pmatrix} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & 1 - e_i^T B^T PBe_i \end{pmatrix} - \tau_i \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

  for all the vertices e_i of the hypercube of dimension p.

+ Previous equations amounts to the smallest ellipsoid such that $\forall x, \forall u, ||u||_\infty \leq 1 \Rightarrow x^T P\, x \leq 1 \Rightarrow (Ax + Bu)^T P\, (Ax + Bu) \leq 1$, i.e. theoretically the "best" solution.

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

$$\begin{pmatrix} -A^T P A & -A^T P B e_i \\ -e_i^T B^T P A & 1 - e_i^T B^T P B e_i \end{pmatrix} - \tau_i \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

  for all the vertices e_i of the hypercube of dimension p.

\+ Previous equations amounts to the smallest ellipsoid such that $\forall x, \forall u, ||u||_\infty \leq 1 \Rightarrow x^T P x \leq 1 \Rightarrow (Ax + Bu)^T P (Ax + Bu) \leq 1$, i.e. theoretically the "best" solution.

– But in practice, this equation is harder to solve (not a LMI).

Heuristic 3: All in One, Approximate Solution

Constraint

$$\begin{pmatrix} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & 1 - e_i^T B^T PBe_i \end{pmatrix} - \tau \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

implies

- $\tau P - A^T PA \succeq 0$, hence $\tau \geq \tau_{min} > 0$ (c.f. r in heuristic 2);

Heuristic 3: All in One, Approximate Solution

Constraint

$$\begin{pmatrix} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & 1 - e_i^T B^T PBe_i \end{pmatrix} - \tau \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

implies

- $\tau P - A^T PA \succeq 0$, hence $\tau \geq \tau_{min} > 0$ (c.f. r in heuristic 2);
- $1 - \left(e_i^T B^T\right) P \left(Be_i\right) - \tau \geq 0$, hence $\tau \leq \tau_{max} < 1$.

Heuristic 3: All in One, Approximate Solution

Constraint

$$\begin{pmatrix} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & 1 - e_i^T B^T PBe_i \end{pmatrix} - \tau \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

implies

- $\tau P - A^T PA \succeq 0$, hence $\tau \geq \tau_{min} > 0$ (c.f. r in heuristic 2);
- $1 - \left(e_i^T B^T\right) P \left(Be_i\right) - \tau \geq 0$, hence $\tau \leq \tau_{max} < 1$.

Method

- Compute τ_{min} and τ_{max};
- Solve LMI for a few equally space values of $\tau \in [\tau_{min}, \tau_{max}]$;
- Keep the best P (maximizing r in $P \succeq rI$).