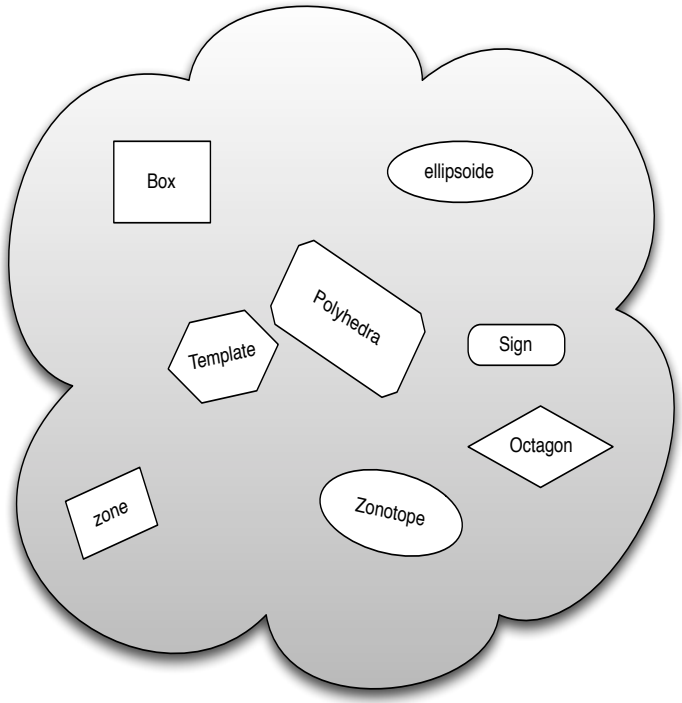
Numerical Abstract Domain using Support Function.

Yassamine Seladji and Olivier Bouissou.

CEA, LIST, LMeASI. France
yassamine.seladji@cea.fr
olivier.bouissou@cea.fr

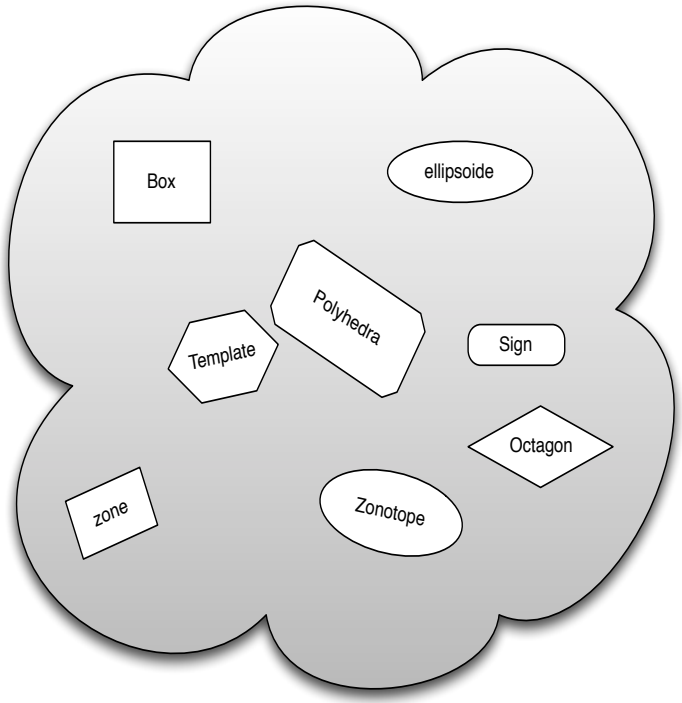
Introduction

Static analysis by abstract interpretation

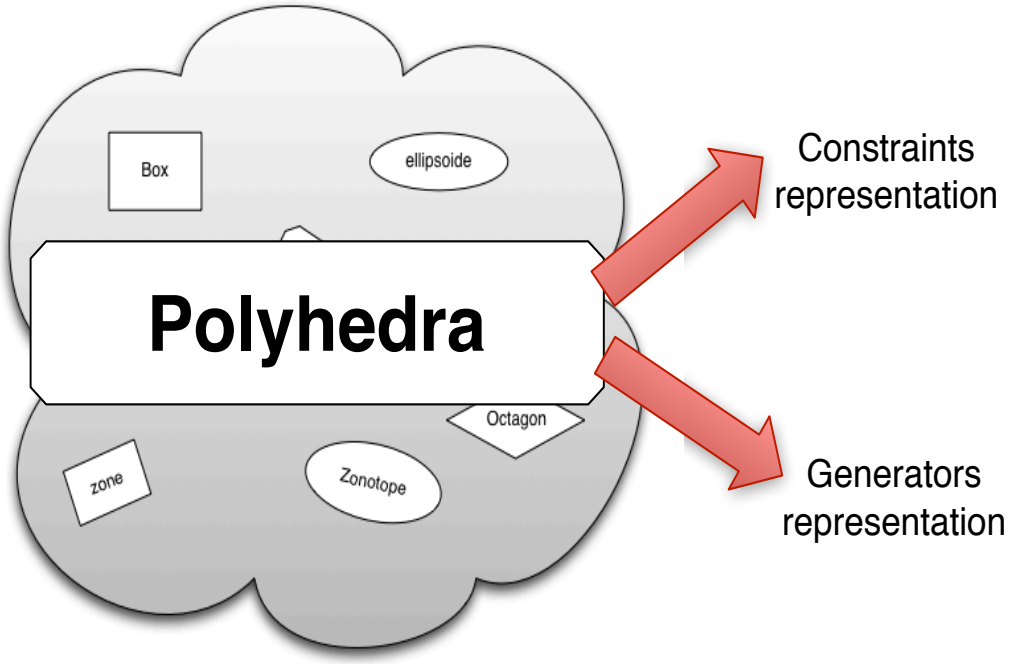


Introduction

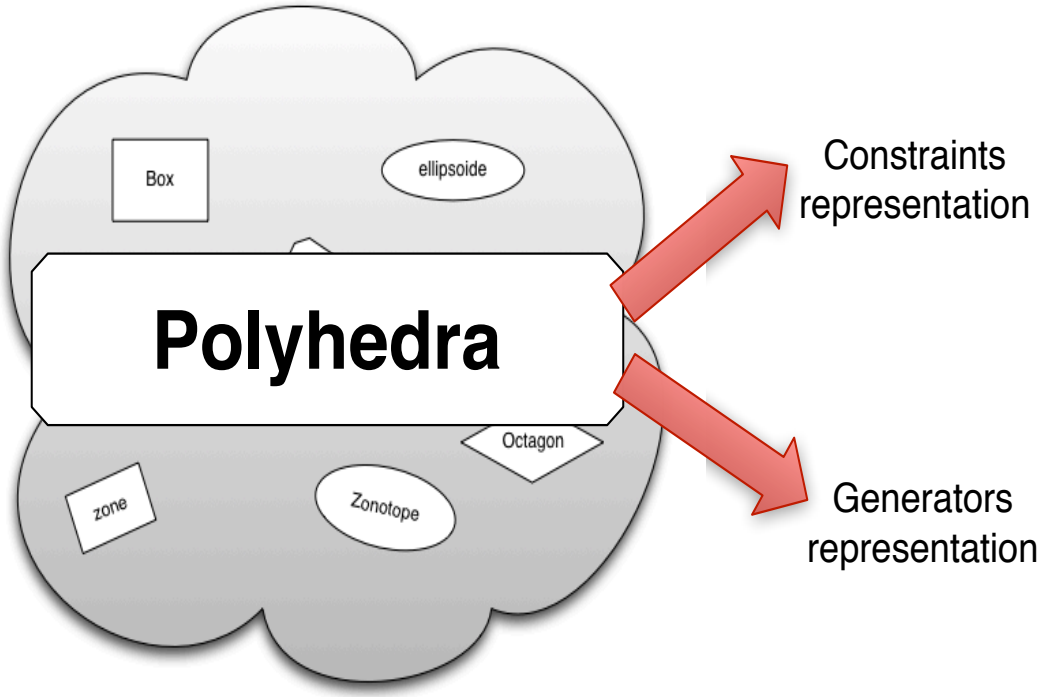
Static analysis by abstract interpretation



Static analysis by abstract interpretation



Static analysis by abstract interpretation

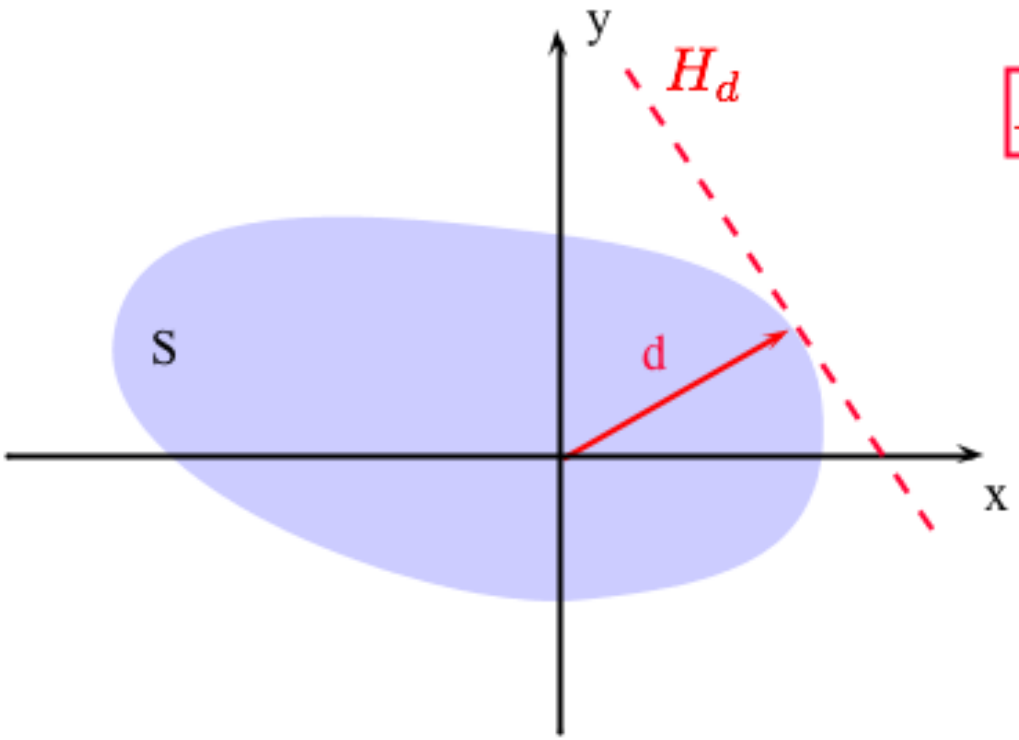


Support function

Definition

Let S be a closed convex set and δ_S its support function, such that :

$$\forall d \in \mathbb{R}^n, \delta_S(d) = sup\{\langle x, d\rangle : x \in S\}$$

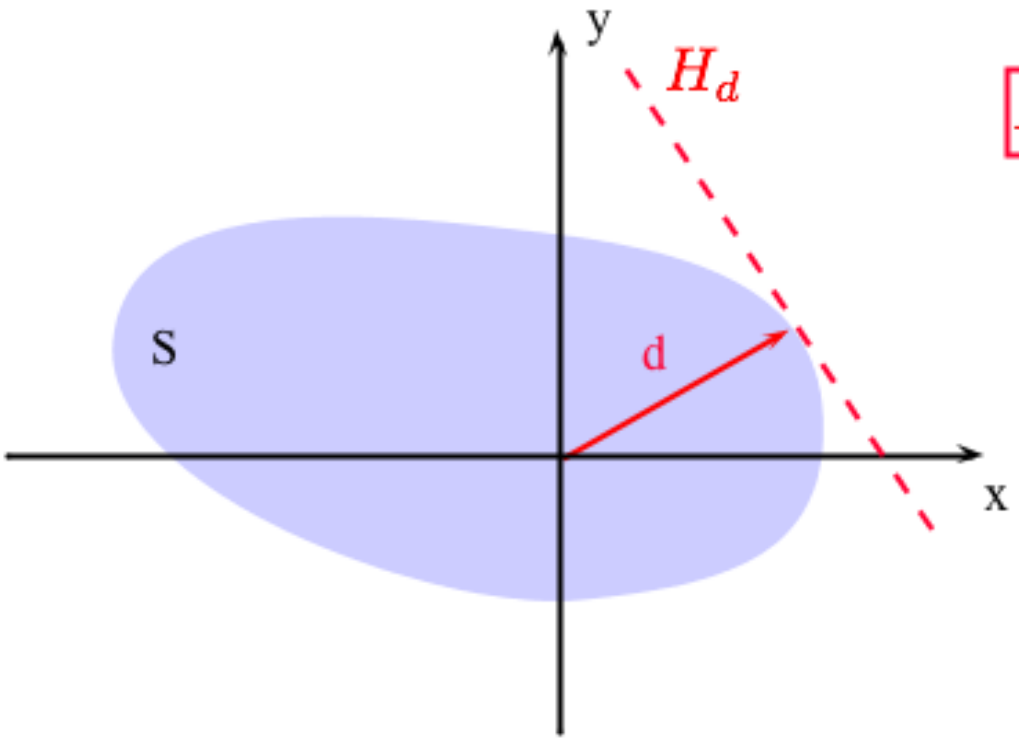


$$H_d : \langle x, d\rangle \leq \delta_S(d)$$

Definition

Let S be a closed convex set and δ_S its support function, such that :

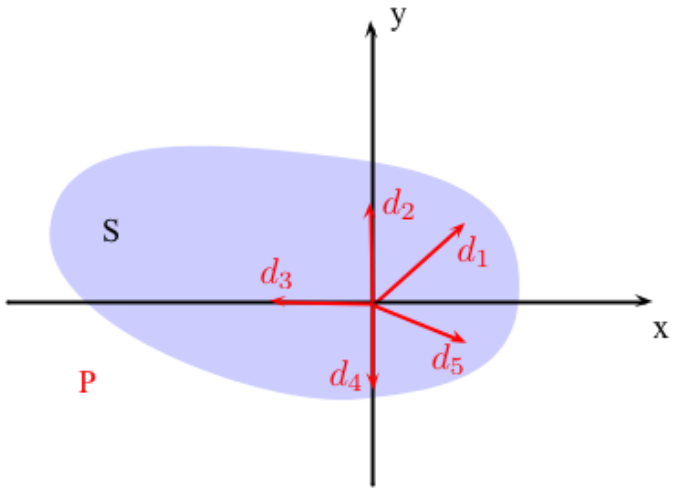
$$\forall d \in \mathbb{R}^n, \delta_S(d) = sup\{\langle x, d\rangle : x \in S\}$$



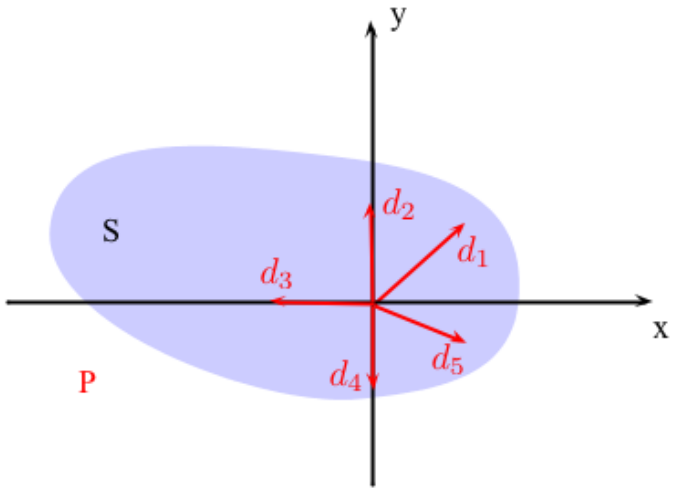
$$H_d : \langle x, d\rangle \leq \delta_S(d)$$

$$S = \cap_{d \in R^n} H_d$$

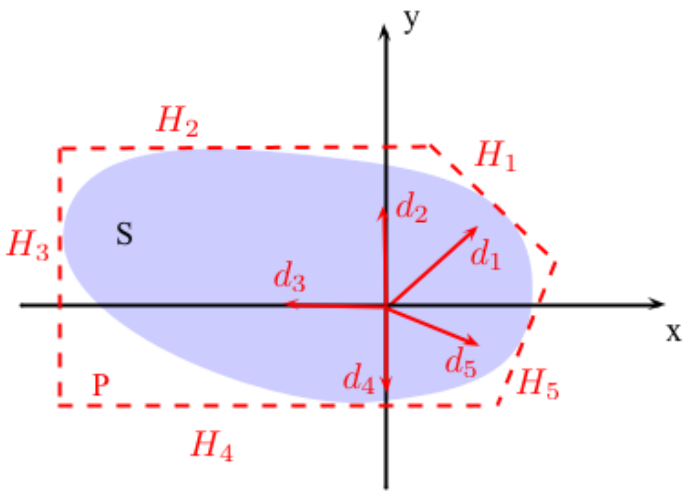
Let $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$ be a set of directions.



Let $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$ be a set of directions.



Let $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$ be a set of directions.



Property

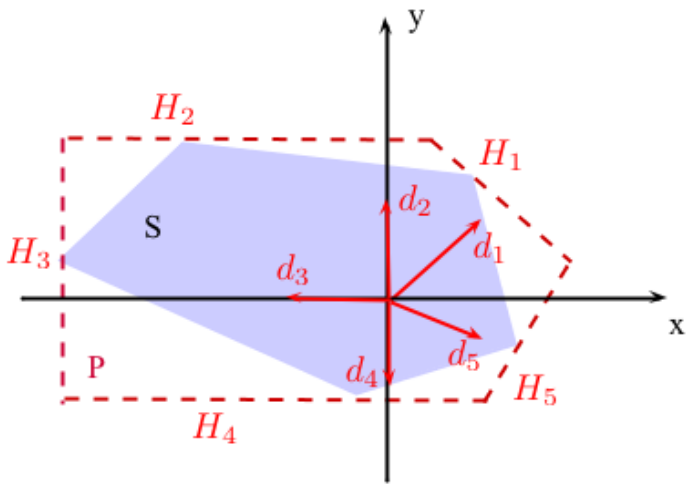
Let S be a closed convex set, and $\Delta \subseteq \mathbb{R}^n$ be a set of directions.
We put

$$P = \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n | \langle x, d \rangle \leq \delta_S(d)\}$$

Then

$$S \subseteq P$$

Let $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$ be a set of directions.



The special case of polyhedron

Let S be a polyhedron. If S is represented by :

- Linear system, δ_S is obtained using Linear Programming.
- Generators (vertices) v_i, $\delta_S(d) = sup\{\langle v_i, d\rangle : v_i \in S\}$.

Properties

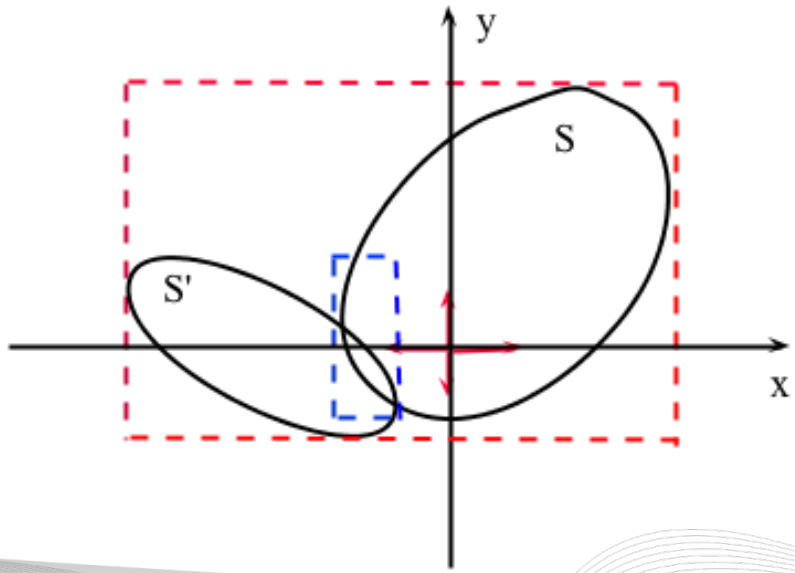
Let S, S' be two closed convex sets. We have :

- $\forall M \in \mathbb{R}^n \times \mathbb{R}^m, \delta_{MS}(d) = \delta_S(M^T d)$.
- $\delta_{S \oplus S'}(d) = \delta_S(d) + \delta_{S'}(d)$. $S \oplus S' = \{x + x' \mid x \in S,\ x' \in S'\}$
- $\delta_{S \cup S'}(d) = \max(\delta_S(d), \delta_{S'}(d))$.
- $\delta_{S \cap S'}(d) \leq \min(\delta_S(d), \delta_{S'}(d))$.

Properties

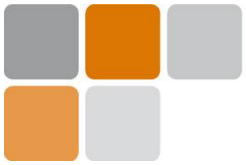
Let S, S' be two closed convex sets. We have :

- $\forall M \in \mathbb{R}^n \times \mathbb{R}^m, \delta_{MS}(d) = \delta_S(M^T d)$.
- $\delta_{S \oplus S'}(d) = \delta_S(d) + \delta_{S'}(d)$. $S \oplus S' = \{x + x' \mid x \in S,\ x' \in S'\}$
- $\delta_{S \cup S'}(d) = \max(\delta_S(d), \delta_{S'}(d))$.
- $\delta_{S \cap S'}(d) \leq \min(\delta_S(d), \delta_{S'}(d))$.

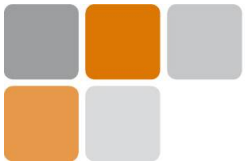


$\lambda d. \max(\delta_S(d), \delta_{S'}(d))$

$\lambda d. \min(\delta_S(d), \delta_{S'}(d))$



For a set of directions Δ,



For a set of directions Δ, let $\mathbb{P}^{\sharp}_{\Delta} = \Delta \to \mathbb{R}_{\infty}$ be the abstract domain.

For a set of directions Δ, let $\mathbb{P}^{\sharp}_{\Delta} = \Delta \to \mathbb{R}_{\infty}$ be the abstract domain.

The concretisation function

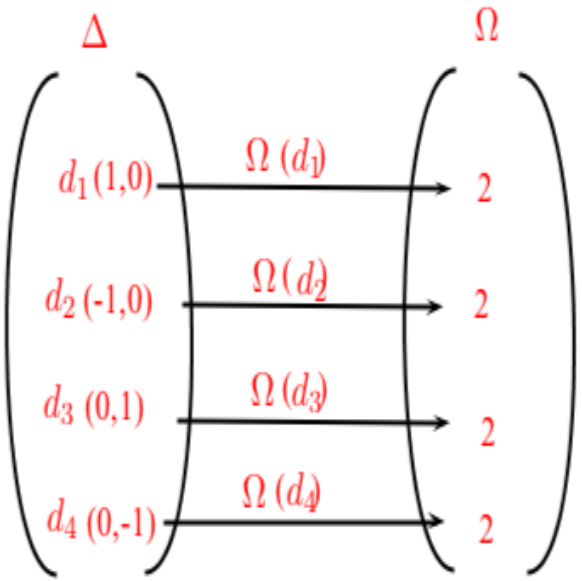
$$\begin{array}{rccl} \gamma_{\Delta} : & (\Delta \to \mathbb{R}_{\infty}) & \longrightarrow & P(\mathbb{R}^n) \\ & \Omega & \longrightarrow & \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\} \end{array}$$

For a set of directions Δ, let $\mathbb{P}^{\sharp}_{\Delta} = \Delta \to \mathbb{R}_{\infty}$ be the abstract domain.

The concretisation function

$$\begin{array}{rccl} \gamma_{\Delta} : & (\Delta \to \mathbb{R}_{\infty}) & \longrightarrow & P(\mathbb{R}^n) \\ & \Omega & \longrightarrow & \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\} \end{array}$$

Example :

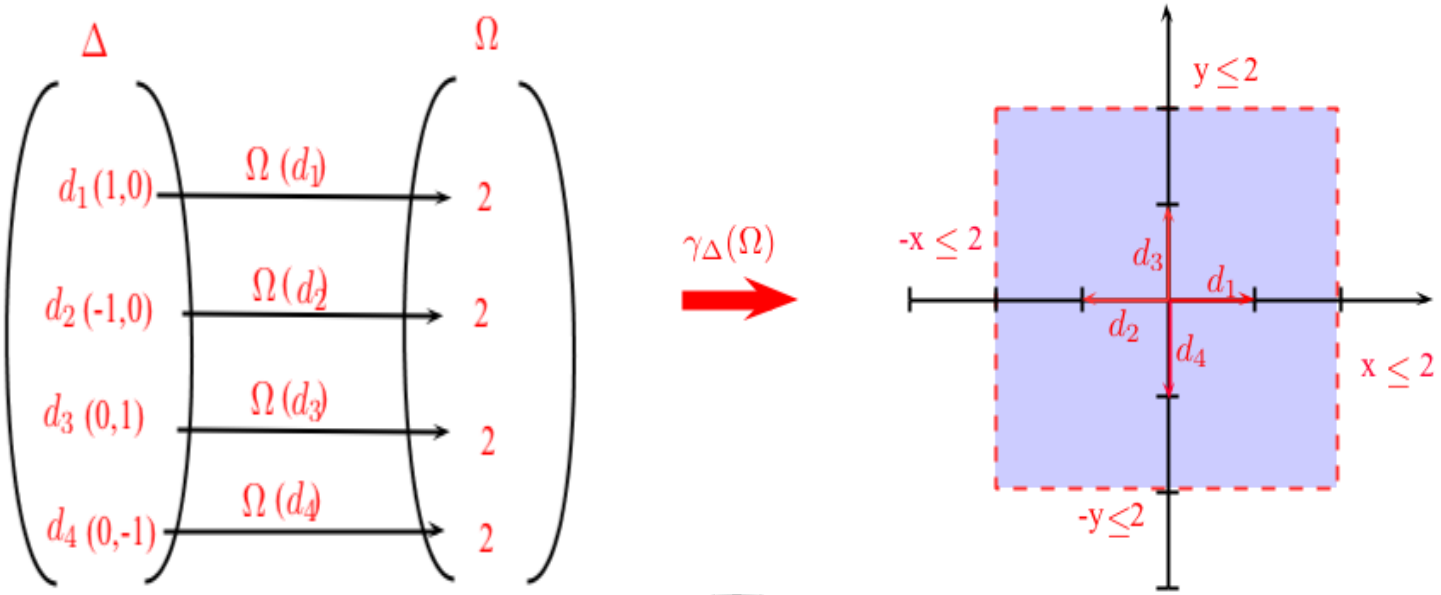


For a set of directions Δ, let $\mathbb{P}^{\sharp}_{\Delta} = \Delta \to \mathbb{R}_{\infty}$ be the abstract domain.

The concretisation function

$$\begin{array}{rccl} \gamma_{\Delta} : & (\Delta \to \mathbb{R}_{\infty}) & \longrightarrow & P(\mathbb{R}^n) \\ & \Omega & \longrightarrow & \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\} \end{array}$$

Example :



The abstraction function

$$\alpha_\Delta : \quad P(\mathbb{R}^n) \longrightarrow (\Delta \to \mathbb{R}_\infty)$$

$$S \longrightarrow \begin{cases} \lambda d.\ -\infty & \text{if } S = \emptyset \\ \lambda d.\ +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d.\ \delta_S(d) & \text{otherwise} \end{cases}$$

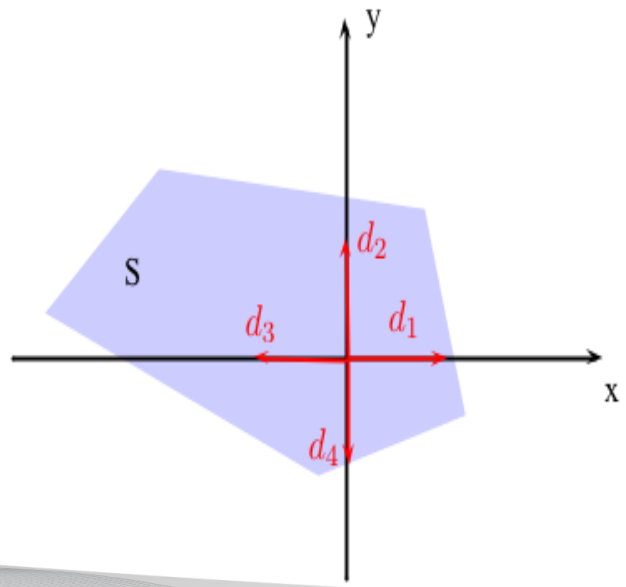
Example :

The abstraction function

$$\alpha_\Delta : \quad P(\mathbb{R}^n) \longrightarrow (\Delta \to \mathbb{R}_\infty)$$

$$S \longrightarrow \begin{cases} \lambda d.\ -\infty & \text{if } S = \emptyset \\ \lambda d.\ +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d.\ \delta_S(d) & \text{otherwise} \end{cases}$$

Example :

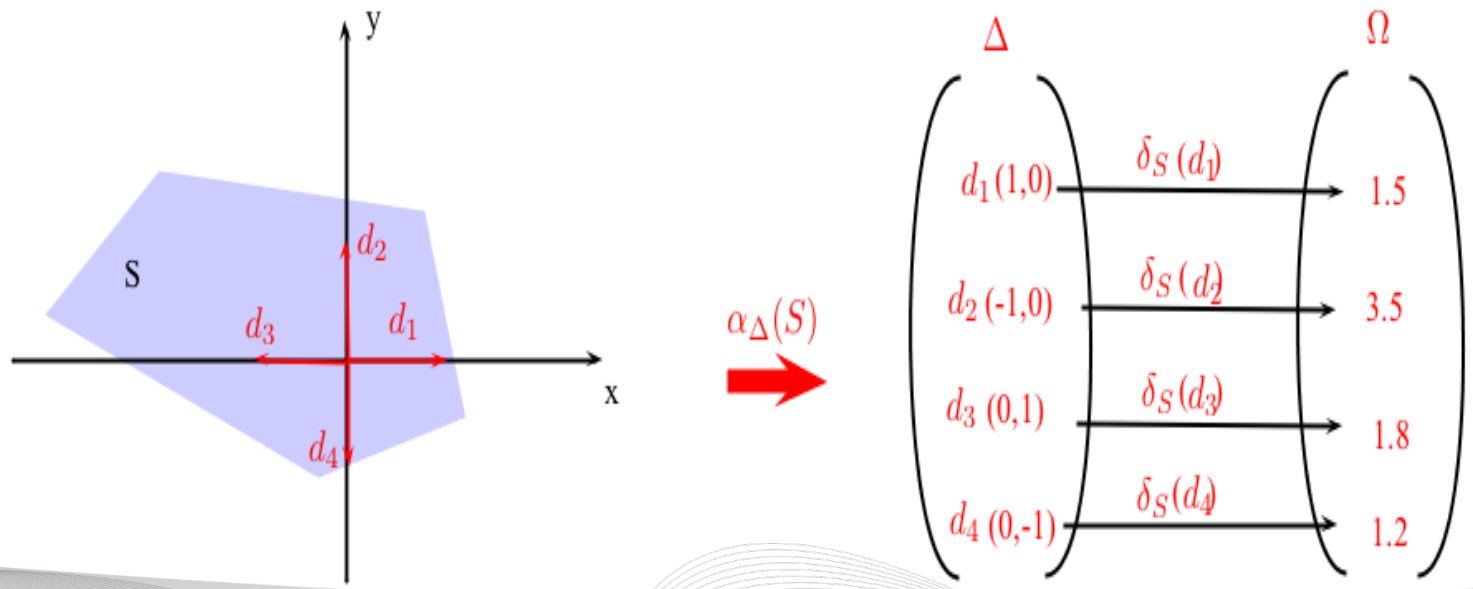


The abstraction function

$$\alpha_\Delta : \quad P(\mathbb{R}^n) \longrightarrow (\Delta \to \mathbb{R}_\infty)$$

$$S \longrightarrow \begin{cases} \lambda d.\ -\infty & \text{if } S = \emptyset \\ \lambda d.\ +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d.\ \delta_S(d) & \text{otherwise} \end{cases}$$

Example :

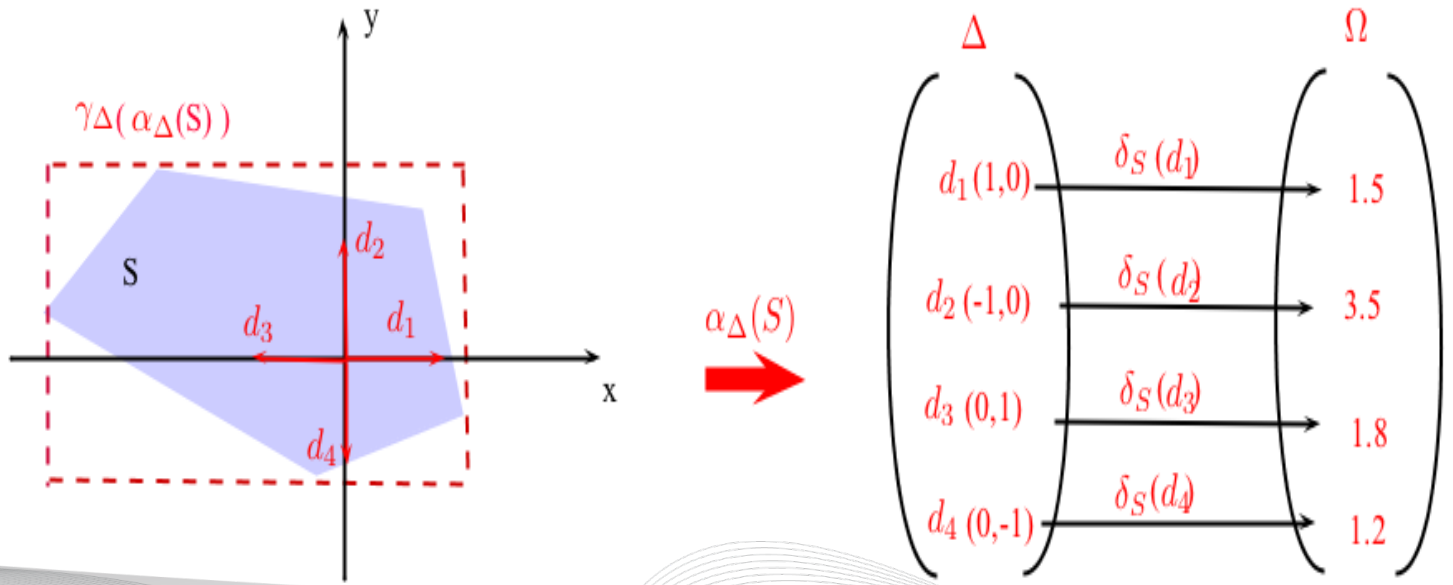


The abstraction function

$$\alpha_\Delta : \quad P(\mathbb{R}^n) \longrightarrow (\Delta \to \mathbb{R}_\infty)$$

$$S \longrightarrow \begin{cases} \lambda d.\ -\infty & \text{if } S = \emptyset \\ \lambda d.\ +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d.\ \delta_S(d) & \text{otherwise} \end{cases}$$

Example :



The complete lattice $\langle \mathbb{P}^{\sharp}_{\Delta}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is defined by :

- An order relation : $\Omega_1 \sqsubseteq \Omega_2 \Leftrightarrow \gamma_{\Delta}(\Omega_1) \subseteq \gamma_{\Delta}(\Omega_2)$.
- A minimal element : $\bot = \lambda d.\ -\infty$.
- A maximal element : $\top = \lambda d.\ +\infty$.
- A join operator : $\Omega_1 \sqcup \Omega_2 = \lambda d.\ \max(\Omega_1(d), \Omega_2(d))$.
- A meet operator : $\Omega_1 \sqcap \Omega_2 = \lambda d.\ \min(\Omega_1(d), \Omega_2(d))$.

The complete lattice $\langle \mathbb{P}^{\sharp}_{\Delta}, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is defined by :

- An order relation : $\Omega_1 \sqsubseteq \Omega_2 \Leftrightarrow \gamma_{\Delta}(\Omega_1) \subseteq \gamma_{\Delta}(\Omega_2)$.
- A minimal element : $\bot = \lambda d.\ -\infty$.
- A maximal element : $\top = \lambda d.\ +\infty$.
- A join operator : $\Omega_1 \sqcup \Omega_2 = \lambda d.\ \max(\Omega_1(d), \Omega_2(d))$.
- A meet operator : $\Omega_1 \sqcap \Omega_2 = \lambda d.\ \min(\Omega_1(d), \Omega_2(d))$.

Notes :

$$\gamma_{\Delta}(\Omega_1 \sqcup \Omega_2) = \gamma_{\Delta}(\Omega_1) \cup \gamma_{\Delta}(\Omega_2).$$
$$\gamma_{\Delta}(\Omega_1 \sqcap \Omega_2) \supseteq \gamma_{\Delta}(\Omega_1) \cap \gamma_{\Delta}(\Omega_2).$$

Program

Input : $\mathbb{P}_0$ a bounded polyhedron.
Input : $A \in \mathbb{R}^n \times \mathbb{R}^m$, $b \in \mathbb{R}^m$.
Input : $c \in \mathbb{R}^n$, $l \in \mathbb{R}$
$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq l)$ {
$\quad X = AX + b.$
}

Program

Input : $\mathbb{P}_0$ a bounded polyhedron.
Input : $A \in \mathbb{R}^n \times \mathbb{R}^m$, $b \in \mathbb{R}^m$.
Input : $c \in \mathbb{R}^n$, $l \in \mathbb{R}$
$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq l)$ {
$X = AX + b.$
}

$$\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$$

- **Case 1 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \# I)]$

Program

$X \in \mathbb{P}_0$
while $(true)$ {
    $X = AX + b$
}

- **Case 1 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \nabla (\Omega_{i-1} \cap I)]$

Program

$X \in \mathbb{P}_0$
while $(true)$ {
  $X = AX + b$
}

The first abstract element

$\Omega_1 = \lambda d.\delta_{\mathbb{P}_0 \cup (A\mathbb{P}_0 \oplus b)}(d)$

- **Case 1 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \nabla (\Omega_{i-1} \cap \{\langle c, X \rangle \leq l\})]$

Program

$X \in \mathbb{P}_0$
while $(true)$ {
  $X = AX + b$
}

The first abstract element

$$\begin{aligned} \Omega_1 &= \lambda d.\delta_{\mathbb{P}_0 \cup (A\mathbb{P}_0 \oplus b)}(d) \\ &= \lambda d.max(\delta_{\mathbb{P}_0}(d), \delta_{\mathbb{P}_0}(A^T d) + \langle b, d \rangle) \end{aligned}$$

Fixpoint computation using Kleene iteration

- **Case 1 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$

Program

$X \in \mathbb{P}_0$
while $(true)$ {
  $X = AX + b$
}

The first abstract element

$$\begin{aligned} \Omega_1 &= \lambda d.\delta_{\mathbb{P}_0 \cup (A\mathbb{P}_0 \oplus b)}(d) \\ &= \lambda d.max(\delta_{\mathbb{P}_0}(d), \delta_{\mathbb{P}_0}(A^T d) + \langle b, d \rangle) \end{aligned}$$

The i^{th} abstract element

$$\Omega_i = \lambda d.\, \max\{\delta_{\mathbb{P}_0}(A^{Tj} d) + \sum_{k=1}^{j} \langle b, A^{T(k-1)} d \rangle, j = 0, .., i\}$$

Fixpoint computation using Kleene iteration

- **Case 1 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

Program

$X \in \mathbb{P}_0$
while $(true)$ {
  $X = AX + b$
}

$$\alpha_\Delta(\mathbb{P}_i) = \Omega_i$$

The i^{th} abstract element

$$\Omega_i = \lambda d.\, \max\{\delta_{\mathbb{P}_0}(A^{Tj} d) + \sum_{k=1}^{j} \langle b, A^{T(k-1)} d \rangle, j = 0, .., i\}$$

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$

Program

$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq l)$
{
$\quad X = AX + b$
}

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

Program

$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq I)$
{
  $X = AX + b$
}

$H : \langle X, c \rangle \leq I$

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

Program

$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq I)$
{
  $X = AX + b$
}

$$H : \langle X, c \rangle \leq I$$

$$\delta_H(d) = \begin{cases} I & \text{if } d = \lambda c, \lambda \geq 0 \\ +\infty & \text{otherwise} \end{cases}$$

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$

Program

$X \in \mathbb{P}_0$
while $(\langle X, c \rangle \leq l)$
{
$X = AX + b$
}

$$H : \langle X, c \rangle \leq l$$

$$\delta_H(d) = \begin{cases} l & \text{if } d = \lambda c, \lambda \geq 0 \\ +\infty & \text{otherwise} \end{cases}$$

We put $\Delta \cup \{c\}$:

- $\Delta_1 = \{c\} \cup \{d \in \Delta | d = \lambda c, \lambda \geq 0\}$.
- $\Delta_2 = \Delta \setminus \Delta_1$

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- For $d \in \Delta_2$: we use the same method as for **Case 1**.

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- For $d \in \Delta_2$: we use the same method as for **Case 1**.

$$\Omega_i = \lambda d. \max\{\delta_{\mathbb{P}_0}(A^{Tj}d) + \sum_{k=1}^{j} \langle b, A^{T(k-1)}d \rangle, j = 0, .., i\}$$

- **Case 2 :** $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- For $d \in \Delta_2$: we use the same method as for **Case 1**.

$$\Omega_i = \lambda d. \max\{\delta_{\mathbb{P}_0}(A^{Tj} d) + \sum_{k=1}^{j} \langle b, A^{T(k-1)} d \rangle, j = 0, .., i\}$$

- For $d \in \Delta_1$, we have that :

$$\Omega_i = \lambda d. \max(\delta_{\gamma_\Delta(\Omega_{i-1})}(d), \min(\delta_{\gamma_\Delta(\Omega_{i-1})}(A^T d) + \langle b, d \rangle, l))$$

Such that $\lambda d. \delta_{\mathbb{P}_i}(d) \leq \Omega_i(d)$.

Fixpoint computation using Kleene iteration

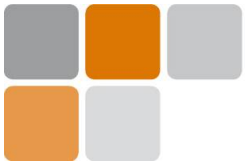
$$\Omega_i(d) = \max\{\delta_{\mathbb{P}_0}(A^{Tj}d) + \sum_{k=1}^{j}\langle b, A^{T(k-1)}d\rangle, j = 0, .., i\}$$

Algorithm 1 Kleene Algorithm using support function.

Require: $\Delta \subset \mathbb{R}^n$, set of l directions. $\mathbb{P}_0$, The initial polyhedron

Require: $A \in \mathbb{R}^n \times \mathbb{R}^m, b \in \mathbb{R}^m$

1: $D = \Delta$, $\Omega = \delta_{\mathbb{P}_0}(\Delta)$
2: **repeat**
3: $\quad\Omega' = \Omega$
4: $\quad$**for all** $i = 0, \ldots, (l-1)$ **do**
5: $\quad\quad\Theta[i] = \Theta[i] + \langle b, D[i]\rangle$
6: $\quad\quad D[i] = A^T D[i]$
7: $\quad\quad\Upsilon[i] = \delta_{\mathbb{P}_0}(D[i]) + \Theta[i]$
8: $\quad\quad\Omega[i] = \max(\Omega[i], \Upsilon[i])$
9: $\quad$**end for**
10: **until** $\Omega \sqsubseteq \Omega'$



The Algorithm doesn't guarantee the termination of the computation.

The Algorithm doesn't guarantee the termination of the computation.

- Solution 1 :

widening

$$\forall \Omega_1, \Omega_2 \in \mathbb{P}^{\sharp}_{\Delta},\ \Omega_1 \nabla_{\Delta} \Omega_2 = \lambda d. \begin{cases} \Omega_2(d) & \text{if } \Omega_1(d) = \Omega_2(d) \\ +\infty & \text{otherwise} \end{cases}$$

The Algorithm doesn't guarantee the termination of the computation.

- Solution 1 :

widening

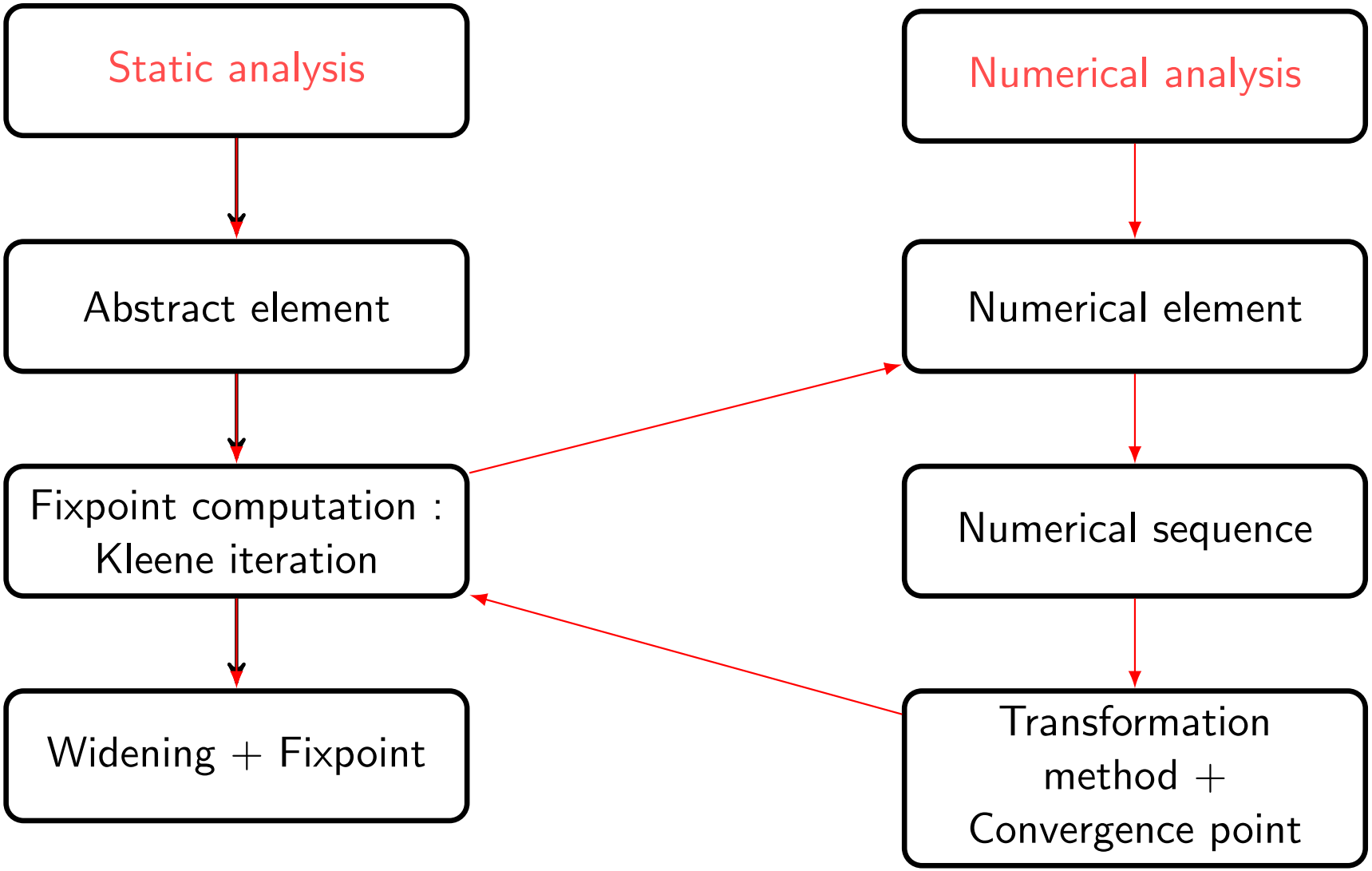
$$\forall \Omega_1, \Omega_2 \in \mathbb{P}^{\sharp}_{\Delta},\ \Omega_1 \nabla_{\Delta} \Omega_2 = \lambda d. \begin{cases} \Omega_2(d) & \text{if } \Omega_1(d) = \Omega_2(d) \\ +\infty & \text{otherwise} \end{cases}$$

- Solution 2 :

The accelerated Kleene iteration

The accelerated Kleene iteration

Seladji and al, JSC 2012



The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000				
1.3333333				
1.2500000				
1.2000000				
1.1666667				
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333				
1.2500000				
1.2000000				
1.1666667				
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000				
1.2000000				
1.1666667				
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000				
1.1666667				
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667				
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667	1.0833333			
1.1428571				
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667	1.0833333			
1.1428571	1.0714287			
1.1250000				
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667	1.0833333			
1.1428571	1.0714287			
1.1250000	1.0624998			
1.1111111				
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667	1.0833333			
1.1428571	1.0714287			
1.1250000	1.0624998			
1.1111111	1.0555557			
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667			
1.2500000	1.1249999			
1.2000000	1.1000001			
1.1666667	1.0833333			
1.1428571	1.0714287			
1.1250000	1.0624998			
1.1111111	1.0555557			
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667	1.1111109		
1.2500000	1.1249999	1.0833337		
1.2000000	1.1000001	1.0666663		
1.1666667	1.0833333	1.0555556		
1.1428571	1.0714287	1.0476161		
1.1250000	1.0624998	1.0416761		
1.1111111	1.0555557			
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667	1.1111109		
1.2500000	1.1249999	1.0833337	1.0624931	
1.2000000	1.1000001	1.0666663	1.0500028	
1.1666667	1.0833333	1.0555556	1.0416545	
1.1428571	1.0714287	1.0476161	1.0357504	
1.1250000	1.0624998	1.0416761		
1.1111111	1.0555557			
1.1000000				

The Aitken-Δ^2 method

Let (S_n) be the initial sequence and (S'_n) its accelerated version

s.t : $\forall n \in, S'_{n+1} = \dfrac{S_{n+1} - S_n}{S_{n+2} - 2S_{n+1} + S_n}$.

Example : $S_n = 1 + \dfrac{1}{n+1}, \forall n \geq$ with $\lim_{n \to +\infty} S_n = 1$

ε_n^0	ε_n^2	ε_n^4	ε_n^6	ε_n^8
2.0000000				
1.5000000	1.2500000			
1.3333333	1.1666667	1.1111109		
1.2500000	1.1249999	1.0833337	1.0624931	
1.2000000	1.1000001	1.0666663	1.0500028	1.0399799
1.1666667	1.0833333	1.0555556	1.0416545	1.0334257
1.1428571	1.0714287	1.0476161	1.0357504	
1.1250000	1.0624998	1.0416761		
1.1111111	1.0555557			
1.1000000				

Accelerated Kleene Algorithm using support function

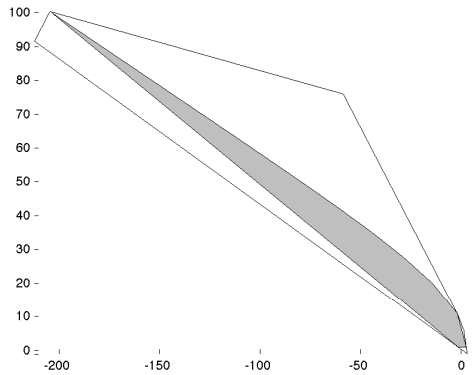
Algorithm 2 Accelerated Kleene Algorithm using support function.

Require: $\Delta \subset \mathbb{R}^n$, $\mathbb{P}_0$, $A \in \mathbb{R}^n \times \mathbb{R}^m$, $b \in \mathbb{R}^m$

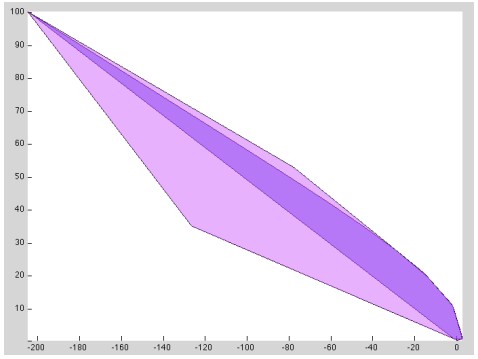
$D = \Delta$, $\Omega = \delta_{\mathbb{P}_0}(\Delta)$
repeat
  $\Omega' = \Omega$, $Y' = Y$
  for all $i = 0, \ldots, (l-1)$ **do**
    $\Theta[i] = \Theta[i] + \langle b, D[i] \rangle$
    $D[i] = A^T D[i]$
    $\Upsilon[i] = \delta_{\mathbb{P}_0}(d[i]) + \Theta[i]$
    $Y[i] = \text{Accelerate}\,(\Upsilon[0], \ldots, \Upsilon[i])$
    if $||Y[i] - Y'[i]|| \leq \varepsilon$ **then**
      $\Omega[i] = \max(\Omega[i], Y[i])$
    else
      $\Omega[i] = \max(\Omega[i], \Upsilon[i])$
    end if
  end for
until $\Omega \sqsubseteq \Omega'$

Kleene Algorithm using support function

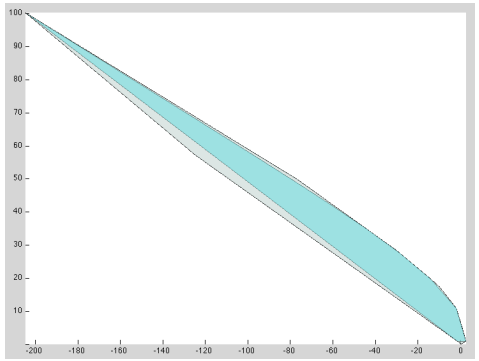
```
begin
    while (0<=10) do
    xn = 0.5 *x - y - 2.5;
     yn = 0.9 *y + 10;
     x = xn; y = yn;
    done;
end
```



8 directions
(0.044 seconds)



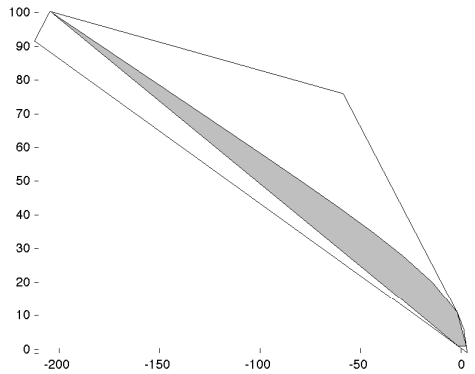
50 directions
(0.34 seconds)



100 directions
(0.7 seconds)

Accelerated Kleene Algorithm using support function

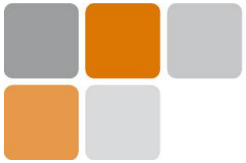
```
begin
    while (0<=10) do
    xn = 0.5 *x - y - 2.5;
     yn = 0.9 *y + 10;
     x = xn; y = yn;
    done;
end
```



8 directions

- Kleene iteration using support function : 200 iterations
- Accelerated Kleene iteration using support function : 11 iterations

1 Le Guernic, C., Girard, A. : Reachability analysis of linear systems using support functions. Nonlinear Analysis : Hybrid Systems (2010)



Related work

1 Le Guernic, C., Girard, A. : Reachability analysis of linear systems using support functions. Nonlinear Analysis : Hybrid Systems (2010)

2 S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In VMCAI. Springer, 2005.

1 Le Guernic, C., Girard, A. : Reachability analysis of linear systems using support functions. Nonlinear Analysis : Hybrid Systems (2010)

2 S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In VMCAI. Springer, 2005.

Similarities :

- Abstract domain based on a static choice of directions set.
- The same definition of inclusion, meet and join operators.

1 Le Guernic, C., Girard, A. : Reachability analysis of linear systems using support functions. Nonlinear Analysis : Hybrid Systems (2010)

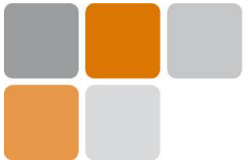
2 S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In VMCAI. Springer, 2005.

Similarities :

- Abstract domain based on a static choice of directions set.
- The same definition of inclusion, meet and join operators.

Differences :

- We have the property that : $\alpha_{\Delta}(\mathbb{P}_i) = \Omega_i$, which is not true for the template abstract domain.
- Our domain reduces the use of linear programming but it's less general than the template abstract domain.



Conclusion and future work

Conclusion

- We develop a new numerical abstract domain based on support function.
- Our abstract domain depends on a set of finite directions.
- In the case where the fixpoint computation terminates, it is obtained in a polynomial time.

Perspectives

- Implements this domain on APRON.
- According to the program to analyse, defines a relevant set of directions.