

# Abstract Interpretation 3/3

David Pichardie

Master recherche Université Rennes 1  
Module PAS



Numeric abstraction by intervals

## Outline

- 1 Numeric abstraction by intervals
- 2 Widening/Narrowing
- 3 Reduced product
- 4 Polyhedral abstract interpretation
- 5 References



# Abstraction by intervals

$$\text{Int} \stackrel{\text{def}}{=} \{ [a, b] \mid a, b \in \overline{\mathbb{Z}}, a \leq b \} \cup \{\perp\} \quad \text{with } \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$$

Lattice :

$$\frac{I \in \text{Int}}{\perp \sqsubseteq_{\text{Int}} I} \quad \frac{c \leq a \quad b \leq d \quad a, b, c, d \in \overline{\mathbb{Z}}}{[a, b] \sqsubseteq_{\text{Int}} [c, d]}$$

$$I \sqcup_{\text{Int}} \perp \stackrel{\text{def}}{=} I, \forall I \in \text{Int}$$

$$\perp \sqcup_{\text{Int}} I \stackrel{\text{def}}{=} I, \forall I \in \text{Int}$$

$$[a, b] \sqcup_{\text{Int}} [c, d] \stackrel{\text{def}}{=} [\min(a, c), \max(b, d)]$$

$$I \sqcap_{\text{Int}} \perp \stackrel{\text{def}}{=} \perp, \forall I \in \text{Int}$$

$$\perp \sqcap_{\text{Int}} I \stackrel{\text{def}}{=} \perp, \forall I \in \text{Int}$$

$$[a, b] \sqcap_{\text{Int}} [c, d] \stackrel{\text{def}}{=} \rho_{\text{Int}}([\max(a, c), \min(b, d)])$$



with  $\rho_{\text{Int}} \in (\overline{\mathbb{Z}} \times \overline{\mathbb{Z}}) \rightarrow \text{Int}$  defined by

$$\rho_{\text{Int}}(a, b) = \begin{cases} [a, b] & \text{if } a \leq b, \\ \perp & \text{otherwise} \end{cases}$$

$$\perp_{\text{Int}} \stackrel{\text{def}}{=} \perp$$

$$\top_{\text{Int}} \stackrel{\text{def}}{=} [-\infty, +\infty]$$

$$\alpha_{\text{Int}}(S) \stackrel{\text{def}}{=} \perp \quad \text{if } S = \emptyset$$

$$\alpha_{\text{Int}}(S) \stackrel{\text{def}}{=} [\min(S), \max(S)] \quad \text{otherwise}$$

$$\gamma_{\text{Int}}(\perp) \stackrel{\text{def}}{=} \emptyset$$

$$\gamma_{\text{Int}}([a, b]) \stackrel{\text{def}}{=} \{ z \in \mathbb{Z} \mid a \leq z \text{ and } z \leq b \}$$



All the other operators are *stricts* : they return  $\perp$  if one of their arguments is  $\perp$ .

$$\begin{aligned} +^\# ([a, b], [c, d]) &= [a + c, b + d] \\ -^\# ([a, b], [c, d]) &= [a - d, b - c] \\ \times^\# ([a, b], [c, d]) &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \end{aligned}$$

$$\llbracket + \rrbracket_{\downarrow \text{op}}^\# ([a, b], [c, d], [e, f]) = (\rho(\max(c, a - f), \min(d, b - e)), \rho(\max(e, a - d), \min(f, b - c)))$$

$$\llbracket - \rrbracket_{\downarrow \text{op}}^\# ([a, b], [c, d], [e, f]) = (\rho(\max(c, a + e), \min(d, b + f)), \rho(\max(e, c - b), \min(f, d - a)))$$

$$\llbracket * \rrbracket_{\downarrow \text{op}}^\# ([a, b], [c, d], [e, f]) = ([c, d], [e, f])$$

$$\llbracket = \rrbracket_{\downarrow \text{comp}}^\# ([a, b], [c, d]) = ([a, b] \sqcap_{\text{Int}} [c, d], [a, b] \sqcap_{\text{Int}} [c, d])$$

$$\llbracket < \rrbracket_{\downarrow \text{comp}}^\# ([a, b], [c, d]) = ([a, b] \sqcap_{\text{Int}} [-\infty, d - 1], [a + 1, +\infty] \sqcap_{\text{Int}} [c, d])$$

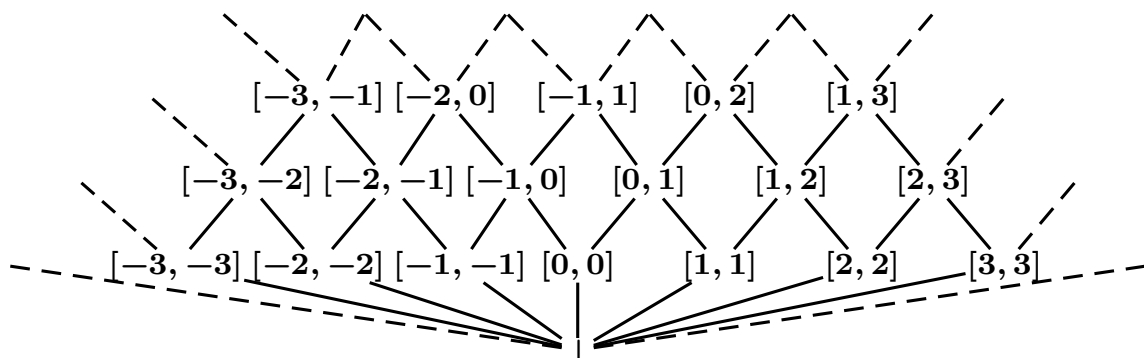
$$\llbracket \leq \rrbracket_{\downarrow \text{comp}}^\# ([a, b], [c, d]) = ([a, b] \sqcap_{\text{Int}} [-\infty, d], [a, +\infty] \sqcap_{\text{Int}} [c, d])$$

$$\llbracket \neq \rrbracket_{\downarrow \text{comp}}^\# ([a, b], [c, d]) = ? \textit{ exercise...}$$

$$\text{const}(n)^\# = [n, n]$$



## Convergence problem



Such a lattice does not satisfy the ascending chain condition.

Example of infinite increasing chain :

$$\perp \sqsubset [0, 0] \sqsubset [0, 1] \sqsubset \dots \sqsubset [0, n] \sqsubset \dots$$

Solution : dynamic approximation

- ▶ we extrapolate the limit thanks to a **widening operator**  $\nabla$

$$\perp \sqsubset [0, 0] \sqsubset [0, 1] \sqsubset [0, 2] \sqsubset [0, +\infty] = [0, 2] \nabla [0, 3]$$



# Outline

- 1 Numeric abstraction by intervals
- 2 Widening/Narrowing
- 3 Reduced product
- 4 Polyhedral abstract interpretation
- 5 References

# Fixpoint approximation

## Lemma

*Let  $(A, \sqsubseteq, \sqcup, \sqcap)$  a complete lattice and  $f$  a monotone operator on  $A$ . If  $a$  is a post-fixpoint of  $f$  (i.e.  $f(a) \sqsubseteq a$ ), then  $\text{lfp}(f) \sqsubseteq a$ .*

We may want to compute an over-approximation of  $\text{lfp}(f)$  in the following cases :

- ▶ The lattice does not satisfies the ascending chain condition, the iteration  $\perp, f(\perp), \dots, f^n(\perp), \dots$  may never terminates.
- ▶ The ascending chain condition is satisfied but the iteration chain is too long to allow an efficient computation.
- ▶ At last, some abstractions do not verify the « good approximation space » criterion previously described. The underlying lattice is not complete, the limits of the ascending iterations do not necessarily belongs to the abstraction domain.

# Widening

Idea : the standard iteration is of the form

$$x^0 = \perp, x^{n+1} = F(x^n) = x^n \sqcup F(x^n)$$

We will replace it by something of the form

$$y^0 = \perp, y^{n+1} = y^n \nabla F(y^n)$$

such that

- (i)  $(y^n)$  is increasing,
- (ii)  $x^n \sqsubseteq y^n$ , for all  $n$ ,
- (iii) and  $(y^n)$  stabilizes after a finite number of steps.

But we also want a  $\nabla$  operator that is independent of  $F$ .

## Widening : definition

A widening is an operator  $\nabla : L \times L \rightarrow L$  such that

- ▶  $\forall x, x' \in L, x \sqcup x' \sqsubseteq x \nabla x'$  (implies (i) & (ii))
- ▶ If  $x^0 \sqsubseteq x^1 \sqsubseteq \dots$  is an increasing chain, then the increasing chain  $y^0 = x^0, y^{n+1} = y^n \nabla x^{n+1}$  stabilizes after a finite number of steps (implies (iii)).

Usage : we replace  $x^0 = \perp, x^{n+1} = F(x^n)$   
by  $y^0 = \perp, y^{n+1} = y^n \nabla F(y^n)$

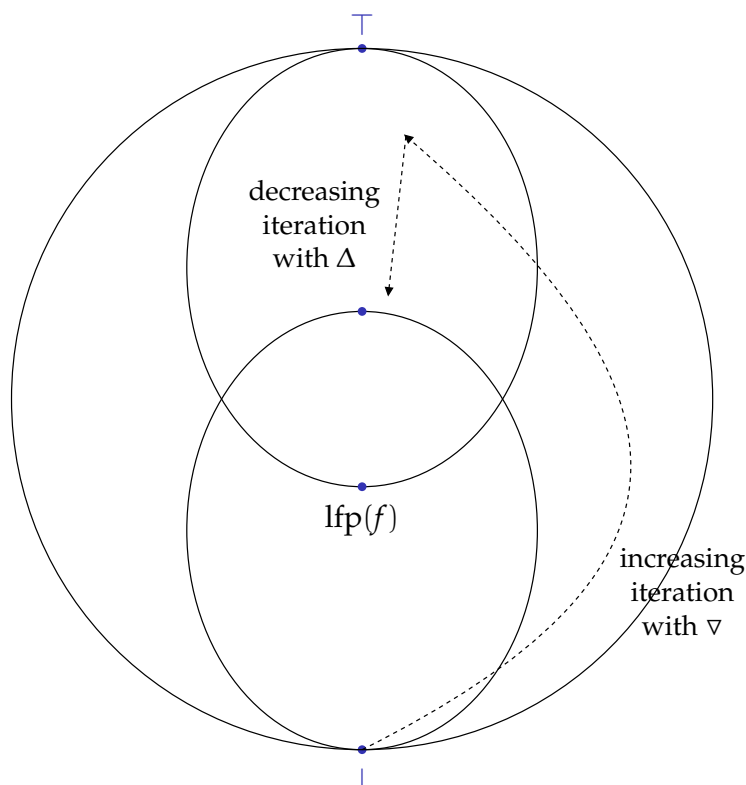
# Widening : theorem

## Theorem

Let  $L$  a complete lattice,  $F : L \rightarrow L$  a monotone function and  $\nabla : L \times L \rightarrow L$  a widening operator. The chain  $y^0 = \perp, y^{n+1} = y^n \nabla F(y^n)$  stabilizes after a finite number of steps towards a post-fixpoint  $y$  of  $F$ .

Corollary :  $\text{lfp}(F) \sqsubseteq y$ .

## Scheme



## Example : widening on intervals

Idea : as soon as a bound is not stable, we extrapolate it by  $+\infty$  (or  $-\infty$ ). After such an extrapolation, the bound can't move any more.

Definition :

$$\begin{aligned}
 [a, b] \nabla_{\text{Int}} [a', b'] &= [ \text{if } a' < a \text{ then } -\infty \text{ else } a, \\
 &\quad \text{if } b' > b \text{ then } +\infty \text{ else } b ] \\
 \perp \nabla_{\text{Int}} [a', b'] &= [a', b'] \\
 I \nabla_{\text{Int}} \perp &= I
 \end{aligned}$$

Examples :

$$[-3, 4] \nabla_{\text{Int}} [-3, 2] = [-3, 4]$$

$$[-3, 4] \nabla_{\text{Int}} [-3, 5] = [-3, +\infty]$$

## Example

```
x := 100;
```

```
while 0 < x {
```

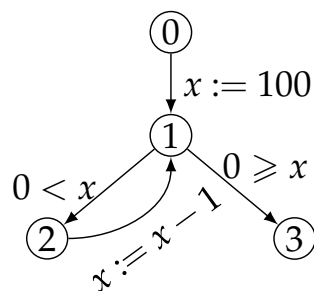
```
    x := x - 1;
```

```
}
```

$$X_1 = [100, 100] \sqcup_{\text{Int}} (X_2 \# [1, 1])$$

$$X_2 = [1, +\infty] \sqcap_{\text{Int}} X_1$$

$$X_3 = [-\infty, 0] \sqcap_{\text{Int}} X_1$$



## Example : without widening

$$\begin{aligned} X_1 &= [100, 100] \sqcup_{\text{Int}} (X_2 -\# [1, 1]) \\ X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\ X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1 \end{aligned}$$

Iteration strategy :  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \dots$

$$\begin{aligned} X_1^0 &= \perp & X_1^{n+1} &= [100, 100] \sqcup_{\text{Int}} (X_2^n -\# [1, 1]) \\ X_2^0 &= \perp & X_2^{n+1} &= [1, +\infty] \sqcap_{\text{Int}} X_1^{n+1} \\ X_3^0 &= \perp & X_3^{n+1} &= [-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1} \end{aligned}$$

$X_1$	$\perp$	$\dots$
$X_2$	$\perp$	$\dots$
$X_3$	$\perp$	$\dots$

## Example : with widening at each nodes of the cfg

$$\begin{aligned} X_1 &= [100, 100] \sqcup_{\text{Int}} (X_2 -\# [1, 1]) \\ X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\ X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1 \end{aligned}$$

Iteration strategy :  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \dots$

$$\begin{aligned} X_1^0 &= \perp & X_1^{n+1} &= X_1^n \nabla_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n -\# [1, 1])) \\ X_2^0 &= \perp & X_2^{n+1} &= X_2^n \nabla_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1}) \\ X_3^0 &= \perp & X_3^{n+1} &= X_3^n \nabla_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}) \end{aligned}$$

$X_1$	$\perp$
$X_2$	$\perp$
$X_3$	$\perp$

## Improving fixpoint approximation

Idea : iterating a little more may help...

### Theorem

Let  $(A, \sqsubseteq, \sqcup, \sqcap)$  a complete lattice,  $f$  a monotone operator on  $A$  and  $a$  a post-fixpoint of  $f$ . The chain  $(x_n)_n$  defined by  $\begin{cases} x_0 &= a \\ x_{k+1} &= f(x_k) \end{cases}$  admits for limit  $(\sqcap \{x_n\})$  the greatest fixpoint of  $f$  lower than  $a$  (written  $\text{gfp}_a(f)$ ). In particular,  $\text{lfp}(f) \sqsubseteq \sqcap \{x_n\}$ . Each intermediate step is a correct approximation :

$$\forall k, \text{lfp}(f) \sqsubseteq \text{gfp}_a(f) \sqsubseteq x_k \sqsubseteq a$$

## Narrowing : definition

A *narrowing* is an operator  $\Delta : L \times L \rightarrow L$  such that

- ▶  $\forall x, x' \in L, x' \sqsubseteq x \Delta x' \sqsubseteq x$
- ▶ If  $x^0 \sqsupseteq x^1 \sqsupseteq \dots$  is a decreasing chain, then the chain  $y^0 = x^0, y^{n+1} = y^n \Delta x^{n+1}$  stabilizes after a finite number of steps.

## Narrowing : decreasing iteration

### Theorem

If  $\Delta$  is a narrowing operator on a poset  $(A, \sqsubseteq)$ , if  $f$  is a monotone operator on  $A$  and  $a$  is a post-fixpoint of  $f$  then the chain  $(x_n)_n$  defined by  $\begin{cases} x_0 & = & a \\ x_{k+1} & = & x_k \Delta f(x_k) \end{cases}$  stabilizes after a finite number of steps on a post-fixpoint of  $f$  lower than  $a$ .

## Narrowing on intervals

$$\begin{aligned}
 [a, b] \Delta_{\text{Int}} [c, d] &= [\text{if } a = -\infty \text{ then } c \text{ else } a ; \text{ if } b = +\infty \text{ then } d \text{ else } b] \\
 I \Delta_{\text{Int}} \perp &= \perp \\
 \perp \Delta_{\text{Int}} I &= \perp
 \end{aligned}$$

Intuition : we only improve infinite bounds.

In practice : a few standard iterations already improve a lot the result that has been obtained after widening...

- ▶ Assignments by constants and conditional guards make the decreasing iterations efficient : they *filter* the (too big) approximations computed by the widening

## Example : with narrowing at each node of the cfg

$$\begin{aligned} X_1 &= [100, 100] \sqcup_{\text{Int}} (X_2 \text{ --\# } [1, 1]) \\ X_2 &= [1, +\infty] \sqcap_{\text{Int}} X_1 \\ X_3 &= [-\infty, 0] \sqcap_{\text{Int}} X_1 \end{aligned}$$

Iteration strategy :  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \dots$

$$\begin{aligned} X_1^0 &= [-\infty, 100] & X_1^{n+1} &= X_1^n \Delta_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n \text{ --\# } [1, 1])) \\ X_2^0 &= [-\infty, 100] & X_2^{n+1} &= X_2^n \Delta_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1}) \\ X_3^0 &= [-\infty, 0] & X_3^{n+1} &= X_3^n \Delta_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}) \end{aligned}$$

$X_1$	$[-\infty, 100]$
$X_2$	$[-\infty, 100]$
$X_3$	$[-\infty, 0]$

## The particular case of an equation system

Consider a system

$$\begin{cases} x_1 &= f_1(x_1, \dots, x_n) \\ &\vdots \\ x_n &= f_n(x_1, \dots, x_n) \end{cases}$$

with  $f_1, \dots, f_n$  monotones.

Standard iteration :

$$\begin{aligned} x_1^{i+1} &= f_1(x_1^i, \dots, x_n^i) \\ x_2^{i+1} &= f_2(x_1^i, \dots, x_n^i) \\ &\vdots \\ x_n^{i+1} &= f_n(x_1^i, \dots, x_n^i) \end{aligned}$$

Standard iteration with widening :

$$\begin{aligned} x_1^{i+1} &= x_1^i \nabla f_1(x_1^i, \dots, x_n^i) \\ x_2^{i+1} &= x_2^i \nabla f_2(x_1^i, \dots, x_n^i) \\ &\vdots \\ x_n^{i+1} &= x_n^i \nabla f_n(x_1^i, \dots, x_n^i) \end{aligned}$$

## The particular case of an equation system

$$\begin{cases} x_1 = f_1(x_1, \dots, x_n) \\ \vdots \\ x_n = f_n(x_1, \dots, x_n) \end{cases}$$

It is sufficient (and generally more precise) to use  $\nabla$  for a selection of index  $W$  such that each dependence cycle in the system goes through at least one point in  $W$ .

$$\forall k = 1..n, x_k^{i+1} = \begin{cases} x_k^i \nabla f_k(x_1^i, \dots, x_n^i) & \text{if } k \in W \\ f_k(x_1^i, \dots, x_n^i) & \text{otherwise} \end{cases}$$

**Chaotic iteration** : at each step, we use only one equation, without forgetting one for ever.

**Delayed widening** : It is generally better to wait a few standard iterations before launching the widenings.

## Outline

- 1 Numeric abstraction by intervals
- 2 Widening/Narrowing
- 3 **Reduced product**
- 4 Polyhedral abstract interpretation
- 5 References

## Reduced product

Suppose we have two Galois connections

$$\left(\mathcal{A}, \sqsubseteq, \sqcup\right) \xleftrightarrow[\alpha_1]{\gamma_1} \left(\mathcal{A}_1^\#, \sqsubseteq_1^\#, \sqcup_1^\#\right) \text{ and } \left(\mathcal{A}, \sqsubseteq, \sqcup\right) \xleftrightarrow[\alpha_2]{\gamma_2} \left(\mathcal{A}_2^\#, \sqsubseteq_2^\#, \sqcup_2^\#\right)$$

to abstract a same concrete domain  $\mathcal{A}$ . We can combine these two abstractions with a new connection

$$\left(\mathcal{A}, \sqsubseteq, \sqcup, \sqcap\right) \xleftrightarrow[\alpha]{\gamma} \left(\mathcal{A}_1^\# \times \mathcal{A}_2^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#\right)$$

where

$$(a_1^\#, a_2^\#) \sqsubseteq^\# (b_1^\#, b_2^\#) \iff a_1^\# \sqsubseteq_1^\# b_1^\# \wedge a_2^\# \sqsubseteq_2^\# b_2^\#, \quad \forall (a_1^\#, a_2^\#), (b_1^\#, b_2^\#) \in \mathcal{A}_1^\# \times \mathcal{A}_2^\#$$

...

Intuition : we run the two analyses in parallel without collaboration between them.

$$\forall a \in \mathcal{A}, \alpha(a) = (\alpha_1(a), \alpha_2(a)) \quad \forall (a_1^\#, a_2^\#) \in \mathcal{A}_1^\# \times \mathcal{A}_2^\#, \gamma(a_1^\#, a_2^\#) = \gamma_1(a_1^\#) \sqcap \gamma_2(a_2^\#)$$

We can do better because the new connection is often not a Galois insertion, even if  $(\alpha_1, \gamma_1)$  and  $(\alpha_2, \gamma_2)$  are.



## Example

If we combine the Galois insertion of signs

$$\left(\mathcal{P}(\mathbb{Z}), \subseteq, \cup, \cap\right) \xleftrightarrow[\alpha_1]{\gamma_1} \left(\{\perp_1, 0_1, -1, +1, \top_1\}, \sqsubseteq_1^\#, \sqcup_1^\#, \sqcap_1^\#\right)$$

$$\begin{aligned} \gamma_1(\perp_1) &= \emptyset \\ \gamma_1(0_1) &= \{0\} \\ \gamma_1(-1) &= \mathbb{Z}^- \\ \gamma_1(+1) &= \mathbb{Z}^+ \\ \gamma_1(\top_1) &= \mathbb{Z} \end{aligned}$$

$$\alpha_1(P) = \begin{cases} \perp_1 & \text{if } P = \emptyset \\ 0_1 & \text{if } P = \{0\} \\ -1 & \text{if } P \not\subseteq \{0\} \text{ et } P \subseteq \mathbb{Z}^- \\ +1 & \text{if } P \not\subseteq \{0\} \text{ et } P \subseteq \mathbb{Z}^+ \\ \top_1 & \text{if } P \not\subseteq \mathbb{Z}^- \text{ et } P \not\subseteq \mathbb{Z}^+ \end{cases}$$

with the Galois insertion of parity

$$\left(\mathcal{P}(\mathbb{Z}), \subseteq, \cup, \cap\right) \xleftrightarrow[\alpha_2]{\gamma_2} \left(\{\perp_2, 0_2, 1_2, \top_2\}, \sqsubseteq_2^\#, \sqcup_2^\#, \sqcap_2^\#\right)$$

$$\begin{aligned} \gamma_2(\perp_2) &= \emptyset \\ \gamma_2(0_2) &= \mathbb{Z}_0 \\ \gamma_2(1_2) &= \mathbb{Z}_1 \\ \gamma_2(\top_2) &= \mathbb{Z} \end{aligned}$$

$$\alpha_2(P) = \begin{cases} \perp_2 & \text{if } P = \emptyset \\ 0_2 & \text{if } P \neq \emptyset \text{ et } P \subseteq \mathbb{Z}_0 \\ 1_2 & \text{if } P \neq \emptyset \text{ et } P \subseteq \mathbb{Z}_1 \\ \top_2 & \text{if } P \not\subseteq \mathbb{Z}_0 \text{ et } P \not\subseteq \mathbb{Z}_1 \end{cases}$$

all couples in  $\{(\perp_1, \perp_2), (\perp_1, 0_2), (\perp_1, 1_2), (\perp_1, \top_2), (0_1, \perp_2), (-1, \perp_2), (+1, \perp_2), (\top_1, \perp_2), (0_1, 1_2)\}$  have the same concretization  $\emptyset$ .



## The problem

If we look for a correct approximation of  $\text{succ} : \mathcal{P}(\mathbb{Z}) \rightarrow \mathcal{P}(\mathbb{Z})$  defined by  $\text{succ}(S) = \{x + 1 \mid x \in S\}$ , it is tempting to rely on the optimal approximations  $\text{succ}_1^\# = \alpha_1 \circ \text{succ} \circ \gamma_1$  and  $\text{succ}_2^\# = \alpha_2 \circ \text{succ} \circ \gamma_2$  of each abstractions.

$$\begin{array}{ll}
 \text{succ}_1^\#(\perp_1) & = \perp_1 & \text{succ}_2^\#(\perp_2) & = \perp_2 \\
 \text{succ}_1^\#(0_1) & = +_1 & \text{succ}_2^\#(0_2) & = 1_2 \\
 \text{succ}_1^\#(-_1) & = \top_1 & \text{succ}_2^\#(1_2) & = 0_2 \\
 \text{succ}_1^\#(+_1) & = +_1 & \text{succ}_2^\#(\top_2) & = \top_2 \\
 \text{succ}_1^\#(\top_1) & = \top_1 & & 
 \end{array}$$

Hence we define  $\text{succ}^\# : \mathcal{A}_1^\# \times \mathcal{A}_2^\# \rightarrow \mathcal{A}_1^\# \times \mathcal{A}_2^\#$  by

$$\text{succ}^\#(a_1^\#, a_2^\#) = (\text{succ}_1^\#(a_1^\#), \text{succ}_2^\#(a_2^\#))$$

$\text{succ}^\#$  is always a correct approximation but it is not optimal, why?

## Reduced product

We compute the optimal reduction  $\rho = \alpha \circ \gamma$

$\rho$	$\perp_1$	$0_1$	$-_1$	$+_1$	$\top_1$
$\perp_2$					
$0_2$					
$1_2$					
$\top_2$					

and we make computations on  $\rho(\mathcal{A}_1^\# \times \mathcal{A}_2^\#)$ : the reduced product

# Reduced product

$\rho \circ \text{succ}^\# \circ \rho$  is a correct approximation of  $\text{succ}$ , more precise than  $\text{succ}^\#$ . We have for example, this time,

$$\rho \circ \text{succ}^\# \circ \rho(0_1, 1_2) = (\perp_1, \perp_2) = \alpha \circ \text{succ} \circ \gamma(0_1, 1_2)$$

Hence the reduction operator allows to make « communicate » the abstractions by sign and parity.

However, the situation is not optimal since

$$(-1, 0_2) = \alpha \circ \text{succ} \circ \gamma(-1, 1_2) \sqsubset \rho \circ \text{succ}^\# \circ \rho(-1, 1_2) = (\top_1, 0_2)$$

Conclusion : the reduction operator allows to combine efficiently and *for free* two correct approximations of an operator  $f$ , but in order to obtain an optimal approximation, we need in general, to go back to the definition of  $\alpha \circ f \circ \gamma$ .

## Outline

- 1 Numeric abstraction by intervals
- 2 Widening/Narrowing
- 3 Reduced product
- 4 Polyhedral abstract interpretation**
- 5 References

# Polyhedral abstract interpretation

*Automatic discovery of linear restraints among variables of a program.*  
P. Cousot and N. Halbwachs. POPL'78.



Patrick Cousot



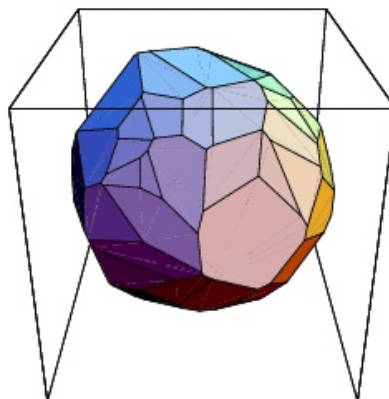
Nicolas Halbwachs

Polyhedral analysis seeks to discover invariant linear equality and inequality relationships among the variables of an imperative program.

## Convex polyhedra

A convex polyhedron can be defined algebraically as the set of solutions of a system of linear inequalities.

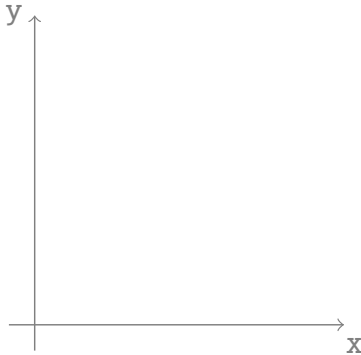
Geometrically, it can be defined as a finite intersection of half-spaces.



# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .

$x = 0; y = 0;$



```

while (x<6) {
  if (?) {

    y = y+2;

  };

  x = x+1;

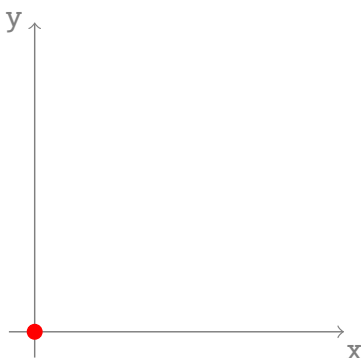
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .

$x = 0; y = 0;$   
 $\{x = 0 \wedge y = 0\}$



```

while (x<6) {
  if (?) {
    {x = 0 \wedge y = 0}
    y = y+2;
  };

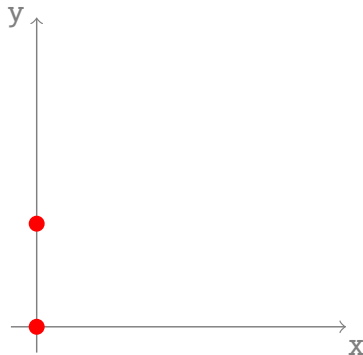
  x = x+1;

}

```

## Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



At junction points, we over-approximates union by a convex union.

```

x = 0; y = 0;
  {x = 0 ∧ y = 0}

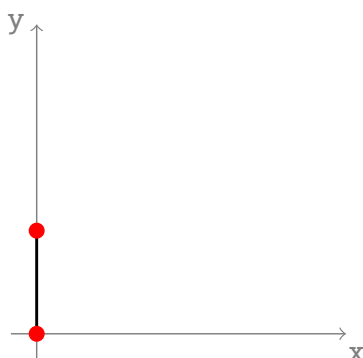
while (x<6) {
  if (?) {
    {x = 0 ∧ y = 0}
    y = y+2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ y = 0} ⊔ {x = 0 ∧ y = 2}

  x = x+1;
}

```

## Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



At junction points, we over-approximates union by a convex union.

```

x = 0; y = 0;
  {x = 0 ∧ y = 0}

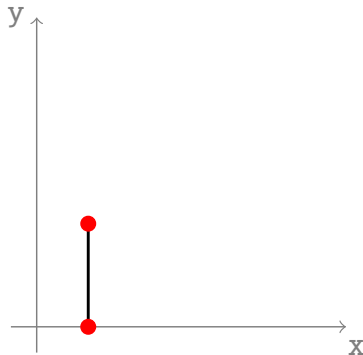
while (x<6) {
  if (?) {
    {x = 0 ∧ y = 0}
    y = y+2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x+1;
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x = 0 ∧ y = 0}

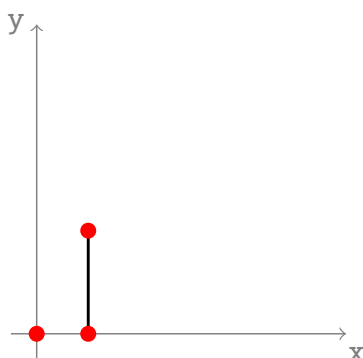
while (x<6) {
  if (?) {
    {x = 0 ∧ y = 0}
    y = y+2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x+1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x = 0 ∧ y = 0} ⊔ {x = 1 ∧ 0 ≤ y ≤ 2}

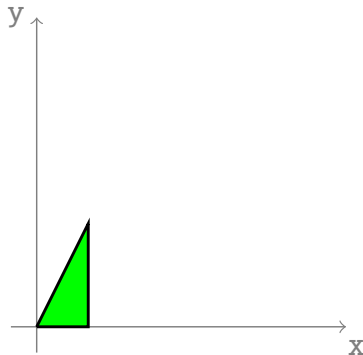
while (x<6) {
  if (?) {
    {x = 0 ∧ y = 0}
    y = y+2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x+1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



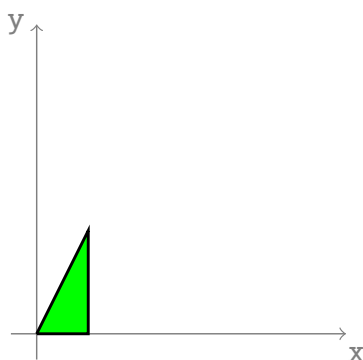
```
x = 0; y = 0;
{x ≤ 1 ∧ 0 ≤ y ≤ 2x}
```

```
while (x < 6) {
  if (?) {
    {x = 0 ∧ y = 0}
    y = y + 2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x + 1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}
```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



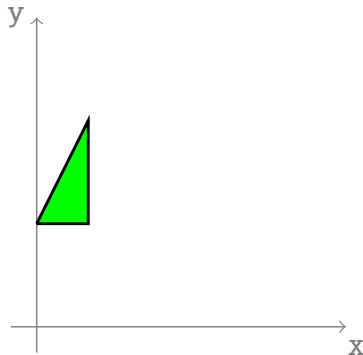
```
x = 0; y = 0;
{x ≤ 1 ∧ 0 ≤ y ≤ 2x}
```

```
while (x < 6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y + 2;
    {x = 0 ∧ y = 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x + 1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}
```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}

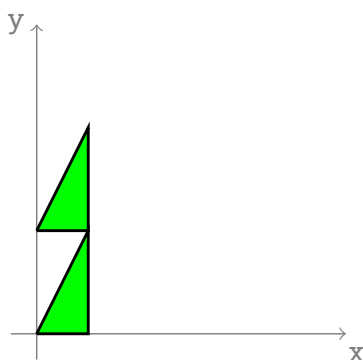
while (x < 6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {x = 0 ∧ 0 ≤ y ≤ 2}

  x = x+1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}

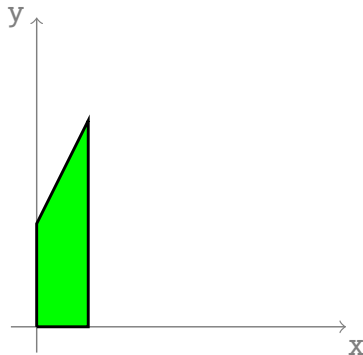
while (x < 6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
  ∪ {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}

  x = x+1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}

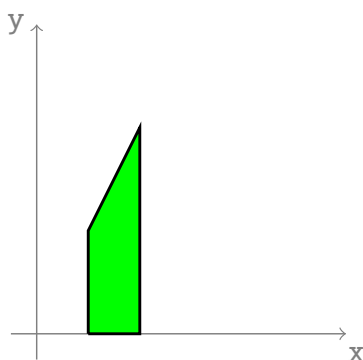
while (x < 6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {0 ≤ x ≤ 1 ∧ 0 ≤ y ≤ 2x + 2}

  x = x+1;
  {x = 1 ∧ 0 ≤ y ≤ 2}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



```

x = 0; y = 0;
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}

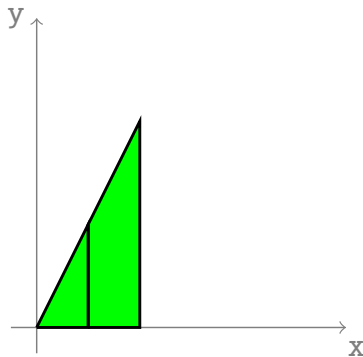
while (x < 6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {0 ≤ x ≤ 1 ∧ 0 ≤ y ≤ 2x + 2}

  x = x+1;
  {1 ≤ x ≤ 2 ∧ 0 ≤ y ≤ 2x}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



At loop headers, we use heuristics (widening) to ensure finite convergence.

```

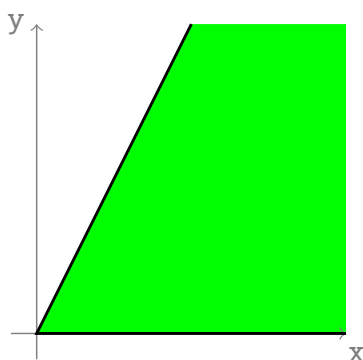
x = 0; y = 0;
  {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
  ∇ {x ≤ 2 ∧ 0 ≤ y ≤ 2x}
while (x<6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {0 ≤ x ≤ 1 ∧ 0 ≤ y ≤ 2x + 2}

x = x+1;
  {1 ≤ x ≤ 2 ∧ 0 ≤ y ≤ 2x}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .



At loop headers, we use heuristics (widening) to ensure finite convergence.

```

x = 0; y = 0;
  {0 ≤ y ≤ 2x}
while (x<6) {
  if (?) {
    {x ≤ 1 ∧ 0 ≤ y ≤ 2x}
    y = y+2;
    {x ≤ 1 ∧ 2 ≤ y ≤ 2x + 2}
  };
  {0 ≤ x ≤ 1 ∧ 0 ≤ y ≤ 2x + 2}

x = x+1;
  {1 ≤ x ≤ 2 ∧ 0 ≤ y ≤ 2x}
}

```

# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .

```

x = 0; y = 0;
  {0 ≤ y ≤ 2x}

while (x < 6) {
  if (?) {
    {0 ≤ y ≤ 2x ∧ x ≤ 5}
    y = y+2;
    {2 ≤ y ≤ 2x + 2 ∧ x ≤ 5}
  };
  {0 ≤ y ≤ 2x + 2 ∧ 0 ≤ x ≤ 5}

  x = x+1;
  {0 ≤ y ≤ 2x ∧ 1 ≤ x ≤ 6}
}
  {0 ≤ y ≤ 2x ∧ 6 ≤ x}

```

By propagation we obtain a post-fixpoint



# Polyhedral analysis

State properties are over-approximated by convex polyhedra in  $\mathbb{Q}^2$ .

```

x = 0; y = 0;
  {0 ≤ y ≤ 2x ∧ x ≤ 6}

while (x < 6) {
  if (?) {
    {0 ≤ y ≤ 2x ∧ x ≤ 5}
    y = y+2;
    {2 ≤ y ≤ 2x + 2 ∧ x ≤ 5}
  };
  {0 ≤ y ≤ 2x + 2 ∧ 0 ≤ x ≤ 5}

  x = x+1;
  {0 ≤ y ≤ 2x ∧ 1 ≤ x ≤ 6}
}
  {0 ≤ y ≤ 2x ∧ 6 = x}

```

By propagation we obtain a post-fixpoint which is enhanced by downward iteration.



# Polyhedral analysis

A more complex example.

```

x = 0; y = A;
  {A ≤ y ≤ 2x + A ∧ x ≤ N}

while (x < N) {
  if (?) {
    {A ≤ y ≤ 2x + A ∧ x ≤ N - 1}
    y = y + 2;
    {A + 2 ≤ y ≤ 2x + A + 2 ∧ x ≤ N - 1}
  };
  {A ≤ y ≤ 2x + A + 2 ∧ 0 ≤ x ≤ N - 1}

  x = x + 1;
  {A ≤ y ≤ 2x + A ∧ 1 ≤ x ≤ N}
}
  {A ≤ y ≤ 2x + A ∧ N = x}

```

The analysis accepts to replace some constants by parameters.



## The four polyhedra operations

- ▶  $\uplus \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$  : convex union
  - ▶ over-approximates the concrete union at junction points
- ▶  $\cap \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$  : intersection
  - ▶ over-approximates the concrete intersection after a conditional instruction
- ▶  $\llbracket x := e \rrbracket \in \mathbb{P}_n \rightarrow \mathbb{P}_n$  : affine transformation
  - ▶ over-approximates the assignment of a variable by a linear expression
- ▶  $\nabla \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$  : widening
  - ▶ ensures (and accelerates) convergence of (post-)fixpoint iteration
  - ▶ includes heuristics to infer loop invariants

```

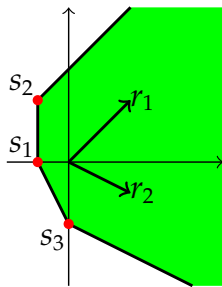
x = 0; y = 0;
  P0 =  $\llbracket y := 0 \rrbracket \llbracket x := 0 \rrbracket (Q^2) \nabla P_4$ 
while (x < 6) {
  if (?) {
    P1 = P0 ∩ {x < 6}
    y = y + 2;
    P2 =  $\llbracket y := y + 2 \rrbracket (P_1)$ 
  };
  P3 = P1 ∪ P2
  x = x + 1;
  P4 =  $\llbracket x := x + 1 \rrbracket (P_3)$ 
}
  P5 = P0 ∩ {x ≥ 6}

```



# Library for manipulating polyhedra

- ▶ Parma Polyhedra Library<sup>1</sup> (PPL), NewPolka : complex C/C++ libraries
- ▶ They rely on the Double Description Method
  - ▶ polyhedra are managed using two representations in parallel



- ▶ by set of inequalities

$$P = \left\{ (x, y) \in \mathbb{Q}^2 \mid \begin{array}{l} x \geq -1 \\ x - y \geq -3 \\ 2x + y \geq -2 \\ x + 2y \geq -4 \end{array} \right\}$$

- ▶ by set of generators

$$P = \left\{ \lambda_1 s_1 + \lambda_2 s_2 + \lambda_3 s_3 + \mu_1 r_1 + \mu_2 r_2 \in \mathbb{Q}^2 \mid \begin{array}{l} \lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2 \in \mathbb{R}^+ \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{array} \right\}$$

- ▶ operations efficiency strongly depends on the chosen representations, so they keep both

1. Previous tutorial on polyhedra partially comes from <http://www.cs.unipr.it/ppl/>

## Outline

- 1 Numeric abstraction by intervals
- 2 Widening/Narrowing
- 3 Reduced product
- 4 Polyhedral abstract interpretation
- 5 References

## References (1)

### A few articles

- ▶ a short formal introduction

*P. Cousot and R. Cousot. Basic Concepts of Abstract Interpretation.*  
<http://www.di.ens.fr/~cousot/COUSOTpapers/WCC04.shtml>

- ▶ technical but very complete (the logic programming part is optional) :

*P. Cousot and R. Cousot. Abstract Interpretation and Application to Logic Programs.*  
<http://www.di.ens.fr/~cousot/COUSOTpapers/JLP92.shtml>

- ▶ a nice application of abstract interpretation theory to verify airbus flight commands

*P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser.*  
<http://www.di.ens.fr/~cousot/COUSOTpapers/ESOP05.shtml>

## References (2)

### On the web :

- ▶ informal presentation of AI with nice pictures

<http://www.di.ens.fr/~cousot/AI/IntroAbsInt.html>

- ▶ a short abstract of various works around AI

<http://www.di.ens.fr/~cousot/AI/>

- ▶ very complete lecture notes

<http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/>