

# Abstract Interpretation

SOS, Master Recherche Science Informatique, U. Rennes 1

2021-2022

Thomas Jensen

(slides by David Pichardie, Thomas Jensen)

# Static program analysis

The goals of static program analysis

- ▶ to prove properties about the run-time behaviour of a program
- ▶ in a fully automatic way
- ▶ without actually executing this program

Abstract interpretation : a theory of **semantic approximation**, which unifies a large variety of static analyses.

Abstract interpretation

- ▶ formalises the **approximated analysis** of programs,
- ▶ allows to **compare the relative precision** of analyses,
- ▶ facilitates **the design** of sophisticated analyses.

# A flavor of abstract interpretation

Abstract interpretation executes programs on **state properties** instead of states.

## Collecting semantics

- ▶ A state property is a subset in  $\mathcal{P}(\mathbb{Z}^2)$  of  $(x, y)$  values.
- ▶ When a point is reached for a second time we make a union with the previous property.
- ▶ We "execute" the program until stability
  - ▶ It may take an infinite number of steps...
  - ▶ But the limit always exists (explained later)

```

x = 0; y = 0;
      {(0,0), (1,0), (1,2), ...}
while (x < 6) {
  if (?) {
      {(0,0), (1,0), (1,2), ...}
    y = y+2;
      {(0,2), (1,2), (1,4), ...}
  };
      {(0,0), (0,2), (1,0), (1,2), (1,4), ...}
x = x+1;
      {(1,0), (1,2), (2,0), (2,2), (2,4), ...}
}
      {(6,0), (6,2), (6,4), (6,6), ...}
  
```

# A flavor of abstract interpretation

Abstract interpretation executes programs on **state properties** instead of states.

## Approximation

- ▶ The set of manipulated properties may be restricted to ensure computability of the semantics.

Example : sign of variables

$$P ::= x \geq 0 \wedge y \geq 0$$

$$C ::= < | \leq | = | > | \geq$$

- ▶ To stay in the domain of selected properties, we over-approximate the concrete properties.

```

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

```

## Another example : the interval analysis

For each point  $k$  and integer variable  $x$ , we infer an interval to which  $x$  *must* belong.

Example : insertion sort, array access verification (buffer overflow)

```

assume(T.length=100); i=1;
                                {i ∈ [1, 100]}
while (i<T.length) {
                                {i ∈ [1, 99]}
    p = T[i]; j = i-1;
                                {i ∈ [1, 99], j ∈ [-1, 98]}
    while (0<=j and T[j]>p) {
                                {i ∈ [1, 99], j ∈ [0, 98]}
        T[j+1]=T[j]; j = j-1;
                                {i ∈ [1, 99], j ∈ [-1, 97]}
    };
                                {i ∈ [1, 99], j ∈ [-1, 98]}
    T[j+1]=p; i = i+1;
                                {i ∈ [2, 100], j ∈ [-1, 98]}
};
                                {i = 100}

```

# Another example : the polyhedral analysis

*For each point  $k$ , infer linear equality and inequality relationships among variables.*

Example : insertion sort, array access verification (buffer overflow)

```
assume(T.length>=1); i=1;
```

$$\{1 \leq i \leq T.length\}$$

```
while i<T.length {
```

$$\{1 \leq i \leq T.length - 1\}$$

```
    p = T[i]; j = i-1;
```

$$\{1 \leq i \leq T.length - 1 \wedge -1 \leq j \leq i - 1\}$$

```
    while 0<=j and T[j]>p {
```

$$\{1 \leq i \leq T.length - 1 \wedge 0 \leq j \leq i - 1\}$$

```
        T[j+1]=T[j]; j = j-1;
```

$$\{1 \leq i \leq T.length - 1 \wedge -1 \leq j \leq i - 2\}$$

```
    };
```

$$\{1 \leq i \leq T.length - 1 \wedge -1 \leq j \leq i - 1\}$$

```
    T[j+1]=p; i = i+1;
```

$$\{2 \leq i \leq T.length + 1 \wedge -1 \leq j \leq i - 2\}$$

```
};
```

$$\{i = T.length\}$$

# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections
- 8 References

# Outline

- 1 Introduction
- 2 Control flow graph**
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections
- 8 References



# While syntax

$Exp ::=$	$n$	$n \in \mathbb{Z}$
	$?$	
	$x$	$x \in \mathbb{V}$
	$Exp \ o \ Exp$	$o \in \{+, -, \times\}$
$test ::=$	$Exp \ c \ Exp$	$c \in \{=, \neq, <, \leq\}$
	$test \ \mathbf{and} \ test$	
	$test \ \mathbf{or} \ test$	
$Stm ::=$	${}^l[x := Exp]$	$l \in \mathbb{P}$
	${}^l[\mathbf{skip}]$	
	$\mathbf{if} \ {}^l[test] \ \{ Stm \} \ \{ Stm \}$	
	$\mathbf{while} \ {}^l[test] \ \{ Stm \}$	
	$Stm ; Stm$	
$Prog ::=$	$[Stm]^{end}$	$end \in \mathbb{P}$

$\mathbb{P}$  : set of program points     $\mathbb{V}$  : set of program variables

# While syntax : example

```
[0[x :=?];  
if 1[x < 0] {  
    while 2[x < 0] {  
        3[x := x + 1];  
    };  
    4[y := x];  
} else {  
    5[y := 0];  
};]6
```

# While semantics

## Semantic domains

$$\begin{aligned} Env &\stackrel{\text{def}}{=} \mathbb{V} \rightarrow \mathbb{Z} \\ State &\stackrel{\text{def}}{=} \mathbb{P} \times Env \end{aligned}$$

## Semantics of expressions

$$\begin{aligned} &\mathcal{A} \llbracket e \rrbracket \rho \in \mathcal{P}(\mathbb{Z}), \quad e \in Exp, \rho \in Env \\ \mathcal{A} \llbracket n \rrbracket \rho &= \{n\} \\ \mathcal{A} \llbracket ? \rrbracket \rho &= \mathbb{Z} \\ \mathcal{A} \llbracket x \rrbracket \rho &= \{ \rho(x) \}, \quad x \in \mathbb{V} \\ \mathcal{A} \llbracket e_1 \circ e_2 \rrbracket \rho &= \{ v_1 \circ v_2 \mid v_1 \in \mathcal{A} \llbracket e_1 \rrbracket \rho, v_2 \in \mathcal{A} \llbracket e_2 \rrbracket \rho \} \\ &\quad \circ \in \{+, -, \times\} \end{aligned}$$

Remark :  $\mathcal{A} \llbracket \cdot \rrbracket \rho$  is non-deterministic because of the expression ?.

## Semantics of tests

$$\mathcal{B} \llbracket t \rrbracket \rho \in \mathcal{P}(\mathbb{B}), \quad t \in \text{test}, \rho \in \text{Env} \quad \mathbb{B} = \{\mathbf{tt}, \mathbf{ff}\}$$

$$\frac{v_1 \in \mathcal{A} \llbracket e_1 \rrbracket \rho \quad v_2 \in \mathcal{A} \llbracket e_2 \rrbracket \rho \quad v_1 \bar{c} v_2}{\mathbf{tt} \in \mathcal{B} \llbracket e_1 \text{ c } e_2 \rrbracket \rho}$$

$$\frac{v_1 \in \mathcal{A} \llbracket e_1 \rrbracket \rho \quad v_2 \in \mathcal{A} \llbracket e_2 \rrbracket \rho \quad \neg(v_1 \bar{c} v_2)}{\mathbf{ff} \in \mathcal{B} \llbracket e_1 \text{ c } e_2 \rrbracket \rho}$$

$$\frac{b_1 \in \mathcal{B} \llbracket t_1 \rrbracket \rho \quad b_2 \in \mathcal{B} \llbracket t_2 \rrbracket \rho}{b_1 \wedge_{\mathbb{B}} b_2 \in \mathcal{B} \llbracket t_1 \text{ and } t_2 \rrbracket \rho}$$

$$\frac{b_1 \in \mathcal{B} \llbracket t_1 \rrbracket \rho \quad b_2 \in \mathcal{B} \llbracket t_2 \rrbracket \rho}{b_1 \vee_{\mathbb{B}} b_2 \in \mathcal{B} \llbracket t_1 \text{ or } t_2 \rrbracket \rho}$$

# Structural Operational Semantics

## Small-step semantics

$$\frac{v \in \mathcal{A}[[a]]\rho}{(l[x := a], \rho) \Rightarrow \rho[x \mapsto v]} \quad \frac{}{(l[\mathbf{skip}], \rho) \Rightarrow \rho}$$

$$\frac{(S_1, \rho) \Rightarrow \rho'}{(S_1 ; S_2, \rho) \Rightarrow (S_2, \rho')} \quad \frac{(S_1, \rho) \Rightarrow (S'_1, \rho')}{(S_1 ; S_2, \rho) \Rightarrow (S'_1 ; S_2, \rho')}$$

$$\frac{\mathbf{tt} \in \mathcal{B}[[b]]\rho}{(\mathbf{if}^l[b] \text{ then } S_1 \text{ else } S_2, \rho) \Rightarrow (S_1, \rho)}$$

$$\frac{\mathbf{ff} \in \mathcal{B}[[b]]\rho}{(\mathbf{if}^l[b] \text{ then } S_1 \text{ else } S_2, \rho) \Rightarrow (S_2, \rho)}$$

$$\frac{\mathbf{tt} \in \mathcal{B}[[b]]\rho}{(\mathbf{while}^l[b] \text{ do } S, \rho) \Rightarrow (S ; \mathbf{while}^l[b] \text{ do } S, \rho)}$$

$$\frac{\mathbf{ff} \in \mathcal{B}[[b]]\rho}{(\mathbf{while}^l[b] \text{ do } S, \rho) \Rightarrow \rho}$$

## Reachable states

**Reachable states** : the set of pairs of program points and states  $(k, \rho)$  such that execution of program  $P$  reaches program point  $k$  with state  $\rho$ .

Formally :

$$\llbracket [P]^{\text{end}} \rrbracket_{\text{SOS}} = \left\{ (k, \rho) \mid \begin{array}{l} \exists \rho_0 \in Env, \\ \exists S \in Stm, (P, \rho_0) \Rightarrow^* (S, \rho) \text{ and } k = \text{entry}(S) \\ \text{or } (P, \rho_0) \Rightarrow^* \rho \text{ and } k = \text{end} \end{array} \right\}$$

# A control flow graph representation of While

The standard program model in static analysis : the *control flow graph*.

The graph model used here :

- ▶ the nodes are program points  $k \in \mathbb{P}$ ,
- ▶ the edges are labeled with *basic instructions*

$$\begin{array}{l} \text{Instr} ::= \quad x := \text{Exp} \quad \text{assignment} \\ \quad \quad | \quad \text{assume test} \quad \text{execution continues only if} \\ \quad \quad \quad \text{the test succeeds} \end{array}$$

- ▶ formally, a cfg is a triple  $(k_{\text{init}}, S, k_{\text{end}})$  with
  - ▶  $k_{\text{init}} \in \mathbb{P}$  : the entry point,
  - ▶  $k_{\text{end}} \in \mathbb{P}$  : the exit point,
  - ▶  $S \subseteq \mathbb{P} \times \text{Instr} \times \mathbb{P}$  the set of edges.

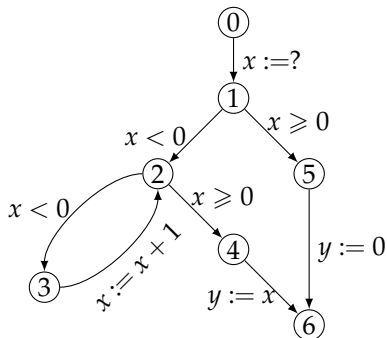
Remark : data-flow analyses are generally based on other versions of control flow graphs (instructions are put in nodes).

# While syntax : example

```

0[x :=?];
if 1[x < 0] {
  while 2[x < 0] {
    3[x := x + 1];
  };
  4[y := x];
} else {
  5[y := 0];
};] 6

```



assume is left implicit



# Small-step semantics of cfg

We first define the semantics of instructions :  $\xrightarrow{i} \subseteq Env \times Env$

$$\frac{v \in \mathcal{A}[[a]]\rho}{\rho \xrightarrow{x := a} \rho[x \mapsto v]} \qquad \frac{\mathbf{tt} \in \mathcal{B}[[t]]\rho}{\rho \xrightarrow{\text{assume } t} \rho}$$

Then, a small-step relation  $\rightarrow_{cfg} \subseteq State \times State$  for a  $cfg = (k_{init}, S, k_{end})$

$$\frac{(k_1, i, k_2) \in S \quad \rho_1 \xrightarrow{i} \rho_2}{(k_1, \rho_1) \rightarrow_{cfg} (k_2, \rho_2)}$$

Reachable states for control flow graphs

$$\llbracket (k_{init}, S, k_{end}) \rrbracket_{CFG} = \{ (k, \rho) \mid \exists \rho_0 \in Env, (k_{init}, \rho_0) \xrightarrow{*}_{(k_{init}, S, k_{end})} (k, \rho) \}$$

# Correctness of $cfg$

## Theorem

For all program  $p$ ,

$$\llbracket cfg(p) \rrbracket_{CFG} = \llbracket p \rrbracket_{SOS}$$

From now on, we write  $\llbracket p \rrbracket$  instead of  $\llbracket cfg(p) \rrbracket_{CFG}$  and we identify  $cfg(p)$  and  $p$ .

# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics**
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections
- 8 References

# Collecting Semantics

We define a **collecting semantics** that gives us the set of reachable states  $\llbracket p \rrbracket_k^{\text{col}}$  at each program point  $k$ .

$$\forall k \in \mathbb{P}, \llbracket p \rrbracket_k^{\text{col}} = \{ \rho \mid (k, \rho) \in \llbracket p \rrbracket \}$$

## Theorem

$\llbracket p \rrbracket^{\text{col}}$  is the least fixpoint of the following equation system.

$$\forall k \in \text{labels}(p), X_k = X_k^{\text{init}} \cup \bigcup_{(k', i, k) \in p} \llbracket i \rrbracket (X_{k'})$$

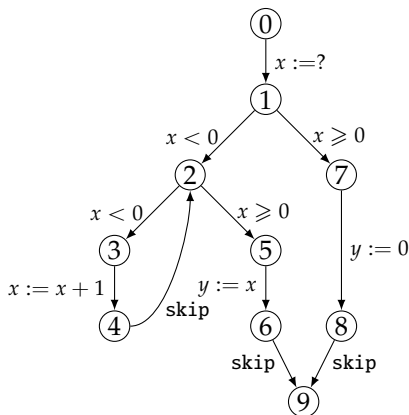
$$\text{with } X_k^{\text{init}} = \begin{cases} \text{Env} & \text{if } k = k_{\text{init}} \\ \emptyset & \text{otherwise} \end{cases}$$

and  $\forall i \in \text{Instr}, \forall X \subseteq \text{Env},$

$$\llbracket i \rrbracket (X) = \left\{ \rho_2 \mid \exists \rho_1 \in X, \rho_1 \xrightarrow{i} \rho_2 \right\}$$

# Example

For the following program,  $\llbracket P \rrbracket^{\text{col}}$  is the least solution of the following equation system :



$$X_0 = Env$$

$$X_1 = \llbracket x := ? \rrbracket (X_0)$$

$$X_2 = \llbracket x < 0 \rrbracket (X_1) \cup X_4$$

$$X_3 = \llbracket x < 0 \rrbracket (X_2)$$

$$X_4 = \llbracket x := x + 1 \rrbracket (X_3)$$

$$X_5 = \llbracket x \geq 0 \rrbracket (X_2)$$

$$X_6 = \llbracket y := x \rrbracket (X_5)$$

$$X_7 = \llbracket x \geq 0 \rrbracket (X_1)$$

$$X_8 = \llbracket y := 0 \rrbracket (X_7)$$

$$X_9 = X_6 \cup X_8$$

## Collecting semantics and exact analysis

The  $(X_k)_{i=1..N}$  are specified as the least solution of a system of equations :

$$X_k = F_k(X_1, X_2, \dots, X_N) \quad , \quad k \in \text{labels}(p)$$

or, equivalently  $\vec{X} = \vec{F}(\vec{X})$ .

Exact analysis :

- ▶ Thanks to Knaster-Tarski, the least solution exists (complete lattice,  $F_k$  are monotone functions),
- ▶ Kleene's fixpoint theorem ( $F_k$  are continuous functions) : it is the limit of

$$X_k^0 = \emptyset \quad , \quad X_k^{n+1} = F_k(X_1^n, X_2^n, \dots, X_N^n)$$

An uncomputable problem :

- ▶ Representing the  $X_k$  may be hard (infinite sets).
- ▶ The limit may not be reachable in a finite number of steps.

# Approximate analysis

Exact analysis :

Least solution of  $X = F(X)$  in the complete lattice  $(\mathcal{P}(Env)^N, \subseteq, \cup, \cap)$ .

Computed as the limit of  $X^0 = \perp, X^{n+1} = F(X^n)$

Approximate analysis :

- ▶ Replace the concrete lattice  $(\mathcal{P}(Env), \subseteq, \cup, \cap)$  by an abstract lattice  $(L^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)$ 
  - ▶ whose elements can be (efficiently) represented,
  - ▶ in which we know how to compute  $\sqcup^\#, \sqcap^\#, \sqsubseteq^\#, \dots$
- ▶ “Transpose” the equation  $X = F(X)$  of  $\mathcal{P}(Env)^N$  into  $X = F^\#(X)$  over  $(L^\#)^N$ .
- ▶ Compute the limit  $X^0 = \perp, X^{n+1} = F^\#(X^n)$

**Fixpoint approximation** : when  $L^\#$  does not verify the ascending chain condition, the iterative computation may not converge in a finite number of steps (or sometimes too slowly).

In this case, we need to approximate the limit (see widening/narrowing).

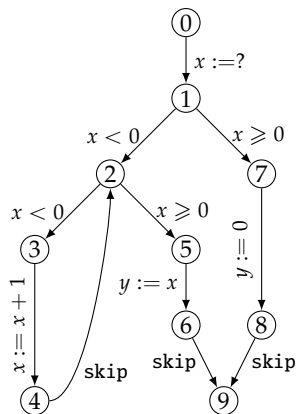
# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation**
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections
- 8 References



Just put some  $\sharp$ ...From  $\mathcal{P}(Env)$  to  $Env^\sharp$ 

control flow graph



collecting semantics

$$\begin{aligned}
 X_0 &= Env \\
 X_1 &= \llbracket x := ? \rrbracket (X_0) \\
 X_2 &= \llbracket x < 0 \rrbracket (X_1) \cup X_4 \\
 X_3 &= \llbracket x < 0 \rrbracket (X_2) \\
 X_4 &= \llbracket x := x + 1 \rrbracket (X_3) \\
 X_5 &= \llbracket x \geq 0 \rrbracket (X_2) \\
 X_6 &= \llbracket y := x \rrbracket (X_5) \\
 X_7 &= \llbracket x \geq 0 \rrbracket (X_1) \\
 X_8 &= \llbracket y := 0 \rrbracket (X_7) \\
 X_9 &= X_6 \cup X_8
 \end{aligned}$$

abstract semantics

$$\begin{aligned}
 X_0^\sharp &= \top_{Env}^\sharp \\
 X_1^\sharp &= \llbracket x := ? \rrbracket^\sharp (X_0^\sharp) \\
 X_2^\sharp &= \llbracket x < 0 \rrbracket^\sharp (X_1^\sharp) \sqcup^\sharp X_4^\sharp \\
 X_3^\sharp &= \llbracket x < 0 \rrbracket^\sharp (X_2^\sharp) \\
 X_4^\sharp &= \llbracket x := x + 1 \rrbracket^\sharp (X_3^\sharp) \\
 X_5^\sharp &= \llbracket x \geq 0 \rrbracket^\sharp (X_2^\sharp) \\
 X_6^\sharp &= \llbracket y := x \rrbracket^\sharp (X_5^\sharp) \\
 X_7^\sharp &= \llbracket x \geq 0 \rrbracket^\sharp (X_1^\sharp) \\
 X_8^\sharp &= \llbracket y := 0 \rrbracket^\sharp (X_7^\sharp) \\
 X_9^\sharp &= X_6^\sharp \sqcup^\sharp X_8^\sharp
 \end{aligned}$$

# Abstract semantics : the ingredients

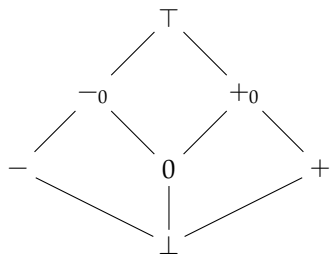
- ▶ A lattice structure  $(Env^\#, \sqsubseteq_{Env^\#}, \sqcup_{Env^\#}, \sqcap_{Env^\#}, \perp_{Env^\#}, \top_{Env^\#})$ 
  - ▶  $\sqsubseteq_{Env^\#}$  is an approximation of  $\subseteq$
  - ▶  $\sqcup_{Env^\#}$  is an approximation of  $\cup$
  - ▶  $\sqcap_{Env^\#}$  is an approximation of  $\cap$
  - ▶  $\perp_{Env^\#}$  is an approximation of  $\emptyset$
  - ▶  $\top_{Env^\#}$  is an approximation of  $Env$
- ▶ For all  $x \in \mathbb{V}, e \in Exp$ , an approximation of  $\llbracket x := e \rrbracket$  :

$$\llbracket x := e \rrbracket^\# \in Env^\# \rightarrow Env^\#$$

- ▶ For all  $t \in test$ , an approximation of  $\llbracket t \rrbracket$  :

$$\llbracket t \rrbracket^\# \in Env^\# \rightarrow Env^\#$$

# An abstraction by signs



$\perp$	represents the property	$\emptyset$
$-$	represents the property	$\{z \mid z < 0\}$
$0$	represents the property	$\{0\}$
$+$	represents the property	$\{z \mid z > 0\}$
$-0$	represents the property	$\{z \mid z \leq 0\}$
$+0$	represents the property	$\{z \mid z \geq 0\}$
$\top$	represents the property	$\mathbb{Z}$

$Env^{\#} \stackrel{\text{def}}{=} \mathbb{V} \rightarrow \mathit{Sign}$  : a sign is associated with each variable.

# An abstraction by signs : example

$$\begin{array}{ll}
 X_0^\sharp & = \top_{Env}^\sharp & X_0^\sharp & = [x : \top; y : \top] \\
 X_1^\sharp & = \llbracket x := ? \rrbracket^\sharp (X_0^\sharp) & X_1^\sharp & = X_0^\sharp[x \mapsto \top] \\
 X_2^\sharp & = \llbracket x < 0 \rrbracket^\sharp (X_1^\sharp) \sqcup^\sharp X_4^\sharp & X_2^\sharp & = X_1^\sharp[x \mapsto -] \sqcup^\sharp X_4^\sharp \\
 X_3^\sharp & = \llbracket x < 0 \rrbracket^\sharp (X_2^\sharp) & X_3^\sharp & = X_2^\sharp[x \mapsto -] \\
 X_4^\sharp & = \llbracket x := x + 1 \rrbracket^\sharp (X_3^\sharp) & X_4^\sharp & = X_3^\sharp[x \mapsto \text{succ}^\sharp(X_3^\sharp(x))] \\
 X_5^\sharp & = \llbracket x \geq 0 \rrbracket^\sharp (X_2^\sharp) & X_5^\sharp & = X_2^\sharp[x \mapsto +_0] \\
 X_6^\sharp & = \llbracket y := x \rrbracket^\sharp (X_5^\sharp) & X_6^\sharp & = X_5^\sharp[y \mapsto X_5^\sharp(x)] \\
 X_7^\sharp & = \llbracket x \geq 0 \rrbracket^\sharp (X_1^\sharp) & X_7^\sharp & = X_1^\sharp[x \mapsto +_0] \\
 X_8^\sharp & = \llbracket y := 0 \rrbracket^\sharp (X_7^\sharp) & X_8^\sharp & = X_7^\sharp[y \mapsto 0] \\
 X_9^\sharp & = X_6^\sharp \sqcup^\sharp X_8^\sharp & X_9^\sharp & = X_6^\sharp \sqcup^\sharp X_8^\sharp
 \end{array}$$

$\xrightarrow[\text{simplifies into}]{\text{which}}$

with

$$\begin{array}{ll}
 \text{succ}^\sharp(\perp) & = \perp \\
 \text{succ}^\sharp(-) & = -_0 \\
 \text{succ}^\sharp(0) & = \text{succ}^\sharp(+) = \text{succ}^\sharp(+_0) = + \\
 \text{succ}^\sharp(-_0) & = \text{succ}^\sharp(\top) = \top
 \end{array}$$

# Exercise

Perform a sign analysis of the program

```
[0[x := -1];  
if 1[x < 0] {  
    2[y := -3];  
} else {  
    3[y := 3];  
};]4
```

- ▶ Generate the cfg.
- ▶ Generate the set of equations.
- ▶ Solve the equation system.
- ▶ Is there a loss of precision? If yes, why?

# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals**
- 6 Widening and narrowing
- 7 Galois connections
- 8 References

# The lattice of intervals

Elements :

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

Order :

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

Lattice operations :

$$\begin{aligned} I \sqcup_{\text{Int}} \perp &\stackrel{\text{def}}{=} I, \forall I \in \text{Int} \\ \perp \sqcup_{\text{Int}} I &\stackrel{\text{def}}{=} I, \forall I \in \text{Int} \\ [a, b] \sqcup_{\text{Int}} [c, d] &\stackrel{\text{def}}{=} [\min(a, c), \max(b, d)] \end{aligned}$$

$$\begin{aligned} I \sqcap_{\text{Int}} \perp &\stackrel{\text{def}}{=} \perp, \forall I \in \text{Int} \\ \perp \sqcap_{\text{Int}} I &\stackrel{\text{def}}{=} \perp, \forall I \in \text{Int} \\ [a, b] \sqcap_{\text{Int}} [c, d] &\stackrel{\text{def}}{=} \rho_{\text{Int}}([\max(a, c), \min(b, d)]) \end{aligned}$$

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

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

Least and greatest element :

$$\begin{aligned} \perp_{\text{Int}} &\stackrel{\text{def}}{=} \perp \\ \top_{\text{Int}} &\stackrel{\text{def}}{=} [-\infty, +\infty] \end{aligned}$$

Abstraction and concretisation :

$$\begin{aligned} \alpha_{\text{Int}}(S) &\stackrel{\text{def}}{=} \perp && \text{if } S = \emptyset \\ \alpha_{\text{Int}}(S) &\stackrel{\text{def}}{=} [\min(S), \max(S)] && \text{otherwise} \end{aligned}$$

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



# Abstraction of basic functions

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

$$+^{\sharp} ([a, b], [c, d]) = [a + c, b + d]$$

$$-^{\sharp} ([a, b], [c, d]) = [a - d, b - c]$$

$$\times^{\sharp} ([a, b], [c, d]) = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

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

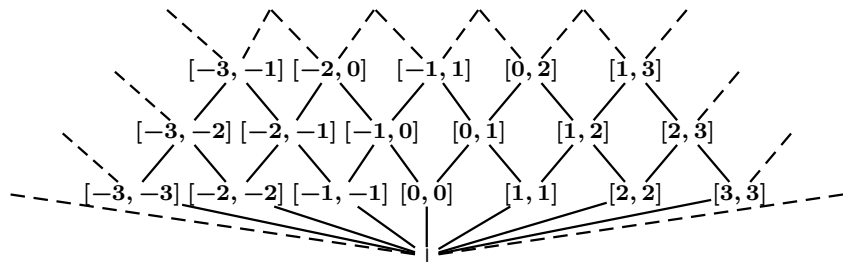
$$\llbracket < \rrbracket_{\downarrow \text{comp}}^{\sharp} ([a, b], [c, d]) = ([a, b] \sqcap_{\text{Int}} [-\infty, d - 1], [a + 1, +\infty] \sqcap_{\text{Int}} [c, d])$$

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

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

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

# Convergence problem



The lattice of intervals 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 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing**
- 7 Galois connections
- 8 References

# Fixpoint approximation

## Lemma

Let  $(A, \sqsubseteq, \sqcup, \sqcap)$  be 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 over-approximate  $\text{lfp}(f)$  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 terminate.
- ▶ The ascending chain condition is satisfied but the iteration chain is too long to allow an efficient computation.
- ▶ The underlying lattice is not complete, so the limits of the ascending iterations do not necessarily belong to the abstraction domain.

# Widening

Idea : the standard iteration is of the form

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

We will replace it by something of the form

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

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

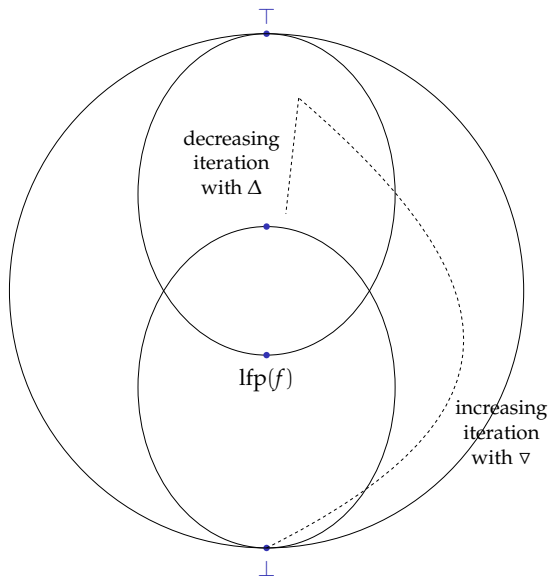
Let

- ▶  $L$  be a complete lattice,
- ▶  $F : L \rightarrow L$  be a monotone function and
- ▶  $\nabla : L \times L \rightarrow L$  a widening operator.

Then, the chain  $y^0 = \perp, y^{n+1} = y^n \nabla F(y^n)$  stabilizes after a finite number of steps at 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

```

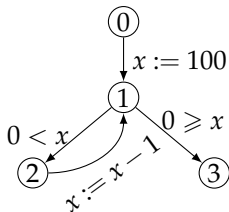
x := 100;
while 0 < x {
  x := x - 1;
}

```

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

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

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



# Example : without widening

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

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

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

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

$$X_1^0 = \perp \quad X_1^{n+1} = [100, 100] \sqcup_{\text{Int}} (X_2^n \text{ -\# } [1, 1])$$

$$X_2^0 = \perp \quad X_2^{n+1} = [1, +\infty] \sqcap_{\text{Int}} X_1^{n+1}$$

$$X_3^0 = \perp \quad X_3^{n+1} = [-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}$$

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

# Example : without widening

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

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

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

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

$$X_1^0 = \perp \quad X_1^{n+1} = [100, 100] \sqcup_{\text{Int}} (X_2^n \text{ -\# } [1, 1])$$

$$X_2^0 = \perp \quad X_2^{n+1} = [1, +\infty] \sqcap_{\text{Int}} X_1^{n+1}$$

$$X_3^0 = \perp \quad X_3^{n+1} = [-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1}$$

$X_1$	$\perp$	$[100, 100]$	$[99, 100]$	$[98, 100]$	$[97, 100]$	$\dots$	$[1, 100]$	$[0, 100]$
$X_2$	$\perp$	$[100, 100]$	$[99, 100]$	$[98, 100]$	$[97, 100]$	$\dots$	$[1, 100]$	$[1, 100]$
$X_3$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\dots$	$\perp$	$[0, 0]$

# Example : with widening at each node of the cfg

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

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

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

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

$$X_1^0 = \perp \quad X_1^{n+1} = X_1^n \nabla_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n \text{ --}^\# [1, 1]))$$

$$X_2^0 = \perp \quad X_2^{n+1} = X_2^n \nabla_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1})$$

$$X_3^0 = \perp \quad X_3^{n+1} = X_3^n \nabla_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1})$$

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

# Example : with widening at each node of the cfg

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

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

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

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

$$X_1^0 = \perp \quad X_1^{n+1} = X_1^n \nabla_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n \text{ --}^\# [1, 1]))$$

$$X_2^0 = \perp \quad X_2^{n+1} = X_2^n \nabla_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1})$$

$$X_3^0 = \perp \quad X_3^{n+1} = X_3^n \nabla_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1})$$

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

# Improving fixpoint approximation

Idea : iterating a little more may help...

## Theorem

Let  $(A, \sqsubseteq, \sqcup, \sqcap)$  be 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  $f$  smaller 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  $(A, \sqsubseteq)$ ,  
and  $f$  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  
at a post-fixpoint of  $f$  lower than  $a$ .

# Narrowing on intervals

$$[a, b] \Delta_{\text{Int}} [c, d] = [\text{if } a = -\infty \text{ then } c \text{ else } a ; \text{ if } b = +\infty \text{ then } d \text{ else } b]$$

$$I \Delta_{\text{Int}} \perp = \perp$$

$$\perp \Delta_{\text{Int}} I = \perp$$

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

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

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

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

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

$$X_1^0 = [-\infty, 100] \quad X_1^{n+1} = X_1^n \Delta_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n \text{ --}^\# [1, 1]))$$

$$X_2^0 = [-\infty, 100] \quad X_2^{n+1} = X_2^n \Delta_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1})$$

$$X_3^0 = [-\infty, 0] \quad X_3^{n+1} = X_3^n \Delta_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1})$$

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

# Example : with narrowing at each node of the cfg

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

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

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

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

$$X_1^0 = [-\infty, 100] \quad X_1^{n+1} = X_1^n \Delta_{\text{Int}} ([100, 100] \sqcup_{\text{Int}} (X_2^n -^\# [1, 1]))$$

$$X_2^0 = [-\infty, 100] \quad X_2^{n+1} = X_2^n \Delta_{\text{Int}} ([1, +\infty] \sqcap_{\text{Int}} X_1^{n+1})$$

$$X_3^0 = [-\infty, 0] \quad X_3^{n+1} = X_3^n \Delta_{\text{Int}} ([-\infty, 0] \sqcap_{\text{Int}} X_1^{n+1})$$

$X_1$	$[-\infty, 100]$	$[-\infty, 100]$	$[0, 100]$
$X_2$	$[-\infty, 100]$	$[1, 100]$	$[1, 100]$
$X_3$	$[-\infty, 0]$	$[-\infty, 0]$	$[0, 0]$

# 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  $W$  **provided** 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.

**Beware** : this time the iteration strategy may affect the precision of the obtained post-fixpoint!

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

# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections**
- 8 References

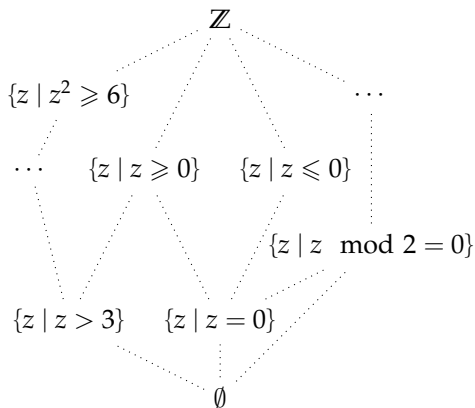
# What is an approximation?. Example : $\mathcal{P}(\mathbb{Z})$

For a program with only one variable, the set of concrete properties is

$$\mathcal{A} = \mathcal{P}(Env) = \mathcal{P}(\mathbb{Z})$$

$$= \left\{ \begin{array}{l} \{z \mid z > 3\}, \\ \{z \mid z^2 \geq 6\}, \\ \{z \mid z \bmod 2 = 0\}, \\ \{z \mid z = 0\}, \\ \{z \mid z \geq 0\}, \\ \dots \end{array} \right\}$$

which can be partially ordered by  $\subseteq$  (logical implication).





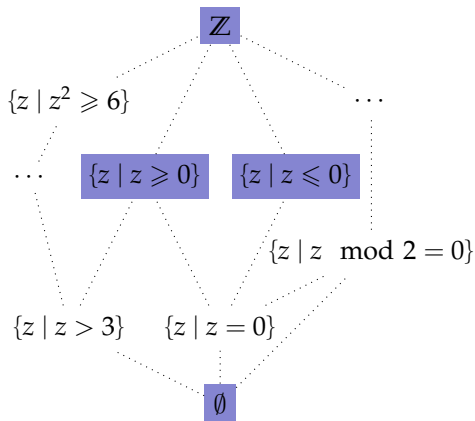
# Example : signs

We select only 4 properties

$$\bar{\mathcal{A}} = \{ \emptyset, \mathbb{Z}^+, \mathbb{Z}^-, \mathbb{Z} \}$$

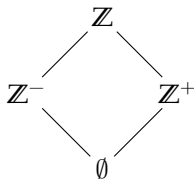
The property  $\ll$  to be greater than 3  $\gg$  (i.e.  $p_3\{z \in \mathbb{Z} \mid z > 3\}$ ) is correctly approximated by  $\mathbb{Z}^+$  et  $\mathbb{Z}$ .

$$p_3 \subseteq \mathbb{Z}^+ \quad p_3 \subseteq \mathbb{Z}$$



## Example : sign analysis

$$\mathcal{A} = \mathcal{P}(\mathbb{Z}) \quad \bar{\mathcal{A}} = \{\emptyset, \mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^-\}$$



The successor operation  $x \mapsto x + 1$  is approximated by

$\bar{p}$	$\emptyset$	$\mathbb{Z}$	$\mathbb{Z}^+$	$\mathbb{Z}^-$
$\overline{\text{succ}(\bar{p})}$	$\emptyset$	$\mathbb{Z}$	$\mathbb{Z}^+$	$\mathbb{Z}$

The predecessor operation  $x \mapsto x - 1$  is approximated by

$\bar{p}$	$\emptyset$	$\mathbb{Z}$	$\mathbb{Z}^+$	$\mathbb{Z}^-$
$\overline{\text{pred}(\bar{p})}$	$\emptyset$	$\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}^-$

Which property of  $\bar{\mathcal{A}}$  do we choose to approximate the property  $\{0\}$ ?

- ▶ if we choose  $\mathbb{Z}^+$

$$x = 0; y = x + 1; z = x - 1 \quad \rightarrow \quad x \in \mathbb{Z}^+, y \in \mathbb{Z}^+, z \in \mathbb{Z}$$

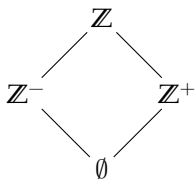
- ▶ if we choose  $\mathbb{Z}^-$

$$x = 0; y = x + 1; z = x - 1 \quad \rightarrow \quad x \in \mathbb{Z}^-, y \in \mathbb{Z}, z \in \mathbb{Z}^-$$

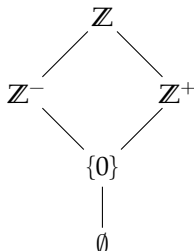
- ▶ however, we would prefer :  $y \in \mathbb{Z}^+, z \in \mathbb{Z}^-$

# Example : sign analysis

$$\mathcal{A} = \mathcal{P}(\mathbb{Z})$$



$$\overline{\mathcal{A}}_1 = \{\emptyset, \mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^-\}$$



$$\overline{\mathcal{A}}_2 = \{\emptyset, \mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^-, \{0\}\}$$

Problem :  $\{0\}$  does not have a best approximation in  $\overline{\mathcal{A}}_1$ .

- ▶ because  $\mathbb{Z}^- \cap \mathbb{Z}^+ \notin \overline{\mathcal{A}}_1$

But  $\overline{\mathcal{A}}_2$  is closed under intersection.

- ▶  $\mathbb{Z}^- \cap \mathbb{Z}^+, \mathbb{Z} \cap \mathbb{Z}^+, \dots, \{0\} \cap \mathbb{Z}^- \cap \mathbb{Z}^+, \dots \in \overline{\mathcal{A}}_2$

# Galois connections

## Definition

Let  $(L_1, \sqsubseteq_1, \sqcup_1, \sqcap_1)$  and  $(L_2, \sqsubseteq_2, \sqcup_2, \sqcap_2)$  be two complete lattices.

A pair of functions  $\alpha \in L_1 \rightarrow L_2$  and  $\gamma \in L_2 \rightarrow L_1$  is a *Galois connection* if it verifies the condition

$$\forall x_1 \in L_1, \forall x_2 \in L_2, \alpha(x_1) \sqsubseteq_2 x_2 \iff x_1 \sqsubseteq_1 \gamma(x_2)$$

# What is a *good* approximation space?

- 1 concrete world : a complete lattice, generally of the form  $(\mathcal{P}(\mathcal{D}), \subseteq, \cup, \cap)$
- 2 abstract world : a complete lattice  $(A^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)$
- 3 link between them : Galois connection.

$$(\mathcal{P}(\mathcal{D}), \subseteq, \cup, \cap) \begin{matrix} \xleftarrow{\gamma} \\ \xrightarrow{\alpha} \end{matrix} (A^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)$$

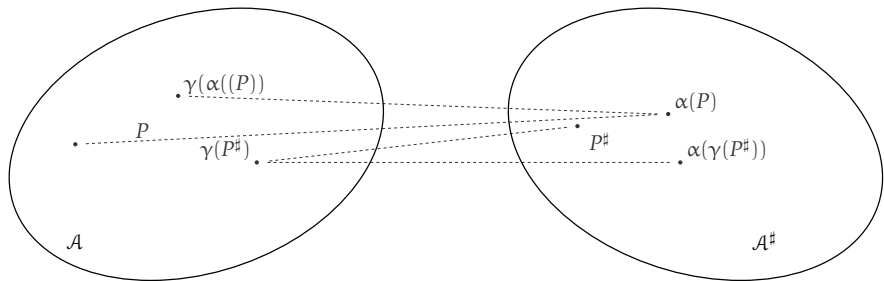
$\ll a^\# \in A^\#$  is a correct approximation of  $a \in \mathcal{P}(\mathcal{D}) \gg$

$$\iff \alpha(a) \sqsubseteq^\# a^\#$$

$$\iff a \subseteq \gamma(a^\#)$$

$\alpha$  : abstraction function       $\gamma$  : concretisation function

Remark : in practice,  $\gamma$  is sufficient to prove the soundness of analyses, but we lose some “nice” theorems...



# Galois connections, properties (1)

If  $\alpha : L \rightarrow L^\sharp, \gamma : L^\sharp \rightarrow L$  is a Galois connection.

- ▶  $\gamma \circ \alpha$  is **extensive** :  $\forall x \in L, x \sqsubseteq \gamma(\alpha(x))$  :  
the abstraction is a correct approximation
- ▶  $\alpha \circ \gamma$  is **retractive** :  $\forall y \in L^\sharp, \alpha(\gamma(y)) \sqsubseteq^\sharp y$  :  
if  $L^\sharp$  is « well chosen »,  $\alpha \circ \gamma = \text{id}$  and (hence)  $\gamma$  is injective (two distinct abstract values never represent the same concrete property)
- ▶  $\alpha$  and  $\gamma$  are monotone  
« the more information we have on a concrete object, the more we have on its abstraction, and *vice-versa* »

Remarks :

- ▶  $x \sqsubseteq x'$  means that  $x$  handles more information than  $x'$ , i.e. gives a more precise information,
- ▶  $\top$  means : everything is possible.

# Function approximation (1)

When some computations in the concrete world are uncomputable or too costly, the abstract world can be used to execute a simplified version of these computations.

- ▶ the abstract computation must always give a conservative answer w.r.t. the concrete computation

Let  $f \in \mathcal{A} \rightarrow \mathcal{A}$  in the concrete world and  $f^\# \in \mathcal{A}^\# \rightarrow \mathcal{A}^\#$  which correctly approximates each concrete computation.

$$\begin{array}{ccc}
 \mathcal{A} & \xrightarrow{f} & \mathcal{A} \\
 \downarrow & & \downarrow \\
 \mathcal{A}^\# & \xrightarrow{f^\#} & \mathcal{A}^\#
 \end{array}$$

$$\alpha(a) \sqsubseteq^\# a^\# \Rightarrow \alpha(f(a)) \sqsubseteq^\# f^\#(a^\#)$$



## Function approximation (1)

If  $\alpha : L \rightarrow L^\sharp, \gamma : L^\sharp \rightarrow L$  is a Galois connection.

### Definition

A function  $f^\sharp \in \mathcal{A}^\sharp \rightarrow \mathcal{A}^\sharp$  is called a *correct approximation* of  $f \in \mathcal{A} \rightarrow \mathcal{A}$  if

$$\forall a \in \mathcal{A}, a^\sharp \in \mathcal{A}^\sharp, \alpha(a) \sqsubseteq^\sharp a^\sharp \Rightarrow \alpha(f(a)) \sqsubseteq^\sharp f^\sharp(a^\sharp)$$

For the monotone abstract functions, we can state several equivalent criteria.

### Theorem

For a monotone function  $f^\sharp \in \mathcal{A}^\sharp \rightarrow \mathcal{A}^\sharp$  and a function  $f \in \mathcal{A} \rightarrow \mathcal{A}$ , the four following assertions are equivalent :

- (i)  $f^\sharp$  is correct approximation of  $f$ ,
- (ii)  $\alpha \circ f \dot{\sqsubseteq}^\sharp f^\sharp \circ \alpha$
- (iii)  $\alpha \circ f \circ \gamma \dot{\sqsubseteq}^\sharp f^\sharp$
- (iv)  $f \circ \gamma \dot{\sqsubseteq} \gamma \circ f^\sharp$

# Fixpoint transfer

## Theorem

Given a Galois connection  $(\mathcal{A}, \sqsubseteq, \sqcup, \sqcap) \xleftrightarrow[\alpha]{\gamma} (\mathcal{A}^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)$ , a monotone function  $f^\# \in \mathcal{A}^\# \rightarrow \mathcal{A}^\#$  and a monotone function  $f \in \mathcal{A} \rightarrow \mathcal{A}$  which verifies  $\alpha \circ f = f^\# \circ \alpha$ , we have

$$\alpha(\text{lfp}(f)) = \text{lfp}(f^\#)$$

## Theorem

Given a Galois connection  $(\mathcal{A}, \sqsubseteq, \sqcup, \sqcap) \xleftrightarrow[\alpha]{\gamma} (\mathcal{A}^\#, \sqsubseteq^\#, \sqcup^\#, \sqcap^\#)$ , a monotone function  $f^\# \in \mathcal{A}^\# \rightarrow \mathcal{A}^\#$  and a monotone function  $f \in \mathcal{A} \rightarrow \mathcal{A}$  which verify  $\alpha \circ f \sqsubseteq^\# f^\# \circ \alpha$ , we have

$$\alpha(\text{lfp}(f)) \sqsubseteq^\# \text{lfp}(f^\#)$$

# Best abstract interpretation

From the equivalences

$$\alpha \circ f \sqsubseteq^{\#} f^{\#} \circ \alpha \iff \alpha \circ f \circ \gamma \sqsubseteq^{\#} f^{\#} \iff f \circ \gamma \sqsubseteq \gamma \circ f^{\#}$$

we have that  $\alpha \circ f \circ \gamma$  is the **best** abstract function  $f^{\#}$  which verifies  $\alpha(\text{lfp}(f)) \sqsubseteq^{\#} \text{lfp}(f^{\#})$ .

**But** :  $\alpha \circ f \circ \gamma$  must not be confused with an implementation of  $f^{\#}$ .

In practice,

- ▶  $\alpha \circ f \circ \gamma$  gives a **specification** of the best abstraction,
- ▶ not an **algorithm** (because  $\alpha$  is rarely computable)

## Exercise

Suppose we analyze programs with 3 variables  $x_1, x_2$  and  $x_3$ . A state is a triple  $(v_1, v_2, v_3)$  where  $v_i \in \mathbb{Z}$  is the current value of variable  $x_i$ .

$$\text{Env} = \mathbb{Z}^3 \quad \mathcal{A} = \mathcal{P}(\mathbb{Z}^3)$$

We would like to design an abstract domain that tracks equalities between variables, and select the following properties for  $\bar{\mathcal{A}}$

$$\bar{\mathcal{A}} = \left\{ \begin{array}{c} \emptyset, \\ \{(v, v, u) \mid v \in \mathbb{Z}, u \in \mathbb{Z}\}, \\ \{(u, v, v) \mid u \in \mathbb{Z}, v \in \mathbb{Z}\}, \\ \{(v, u, v) \mid u \in \mathbb{Z}, v \in \mathbb{Z}\}, \\ \mathbb{Z}^3 \end{array} \right\}$$

What is the lattice (abstract domain) for this selection of properties?

We next want to define the corresponding Galois connection, but that turns out to be difficult—why? What's the problem with our abstract domain? How can it be repaired?

# Exercise

Consider the following concrete operators

$$\llbracket x_1 := ? \rrbracket, \quad \llbracket x_1 := x_2 \rrbracket, \quad \llbracket \text{assume } x_1 = x_2 \rrbracket \in \mathcal{P}(\mathbb{Z}^3) \rightarrow \mathcal{P}(\mathbb{Z}^3)$$

Using the abstraction proposed in the previous exercise, give sound approximations for these operators.

Prove they are sound wrt. the given abstraction choice.

Prove they are optimal wrt. the given abstraction choice.

# Outline

- 1 Introduction
- 2 Control flow graph
- 3 Collecting semantics
- 4 Approximate analysis : an informal presentation
- 5 Abstraction by intervals
- 6 Widening and narrowing
- 7 Galois connections
- 8 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>*

- ▶ application of abstract interpretation 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/>*