# Cours Modèles et ALGorithmes (MALG)
# Cours Modélisation, Vérification et Expérimentation (MOVEX)

# Analyse des programmes (II)

Dominique Méry
Telecom Nancy, Université de Lorraine

**Année universitaire 2022-2023**

# Plan

# Sommaire

# Current Summary

## Position of the problem to solve

- Analysing programs with respect to safety properties
- Computing invariants of a program
- Problem : computing invariants is undecidable
- Idea : developing techniques of abstractions for simplifying computations using abstract interpretation frameworks.

# Current Summary

# Syntax for a Small Programming Language

$$
\begin{array}{lcll}
expr & ::= & v & v \in \mathbb{Z} \\
 & | & ? & \\
 & | & x & x \in \mathbb{V} \\
 & | & expr\ op\ expr & op \in \{+, -, \times, /\} \\
 & & & \\
cond & ::= & expr\ relop\ Expr & relop \in \{<, \leq, >, \geq, =, \\
 & | & \mathbf{not}\ cond & \\
 & | & cond\ \mathbf{and}\ cond & \\
 & & & \\
stmt & ::= & \ell[x := expr] & \ell \in \mathbb{C} \\
 & | & \ell[skip] & \\
 & | & \mathbf{if}\ \ell[cond]\ \mathbf{then}\ stmt\ \mathbf{else}\ stmt\ \mathbf{end\ if} & \\
 & | & \mathbf{while}\ \ell[cond]\ \mathbf{do}\ \ stmt\ \mathbf{end\ do} & \\
 & | & stmt; stmt & \\
actions & ::= & x := exp & \\
 & | & skip & \\
 & | & \textsf{assert}\ cond &
\end{array}
$$

- $\mathbb{C}$ : set of labels for programs.
- $Mem = V \rightarrow \mathbb{Z}$ : set of memory states for variables $V$.
- $\mathcal{E} \in expr \rightarrow (Mem \rightarrow \mathcal{P}(\mathbb{Z}))$ : $\mathcal{E}(e)(s)$ is the set of possible values of $e$ in $s \in Mem$
- $\mathcal{C} \in cond \rightarrow (Mem \rightarrow \mathcal{P}(\mathbb{B}))$ : $\mathcal{C}(cond)(m)$ is the set of possible values of $cond$ in $s \in Mem$.

# Semantics for Languages

Semantics for expressions

$\mathcal{E}[\![v]\!](m) \in \mathcal{P}(\mathbb{Z})$, $e \in Expr$,$m \in Mem$, $x \in \mathbb{V}$, $op \in \{+,-,\times,/\}$

$$\begin{array}{lcl}
\mathcal{E}[\![v]\!](m) & \stackrel{def}{=} & \{v\} \\
\mathcal{E}[\![?]\!](m) & \stackrel{def}{=} & \mathbb{Z} \\
\mathcal{E}[\![x]\!](m) & \stackrel{def}{=} & \{m(x)\} \\
\mathcal{E}[\![e_1 \ op \ e_2]\!](m) & \stackrel{def}{=} & \{v | \exists ve_1, ve_2. \begin{pmatrix} ve_1 \in \mathcal{E}[\![e_1]\!](m) \\ ve_2 \in \mathcal{E}[\![e_2]\!](m) \\ v = ve_1 \ o \ ve_2 \end{pmatrix} \}
\end{array}$$

Semantics for conditions

$\mathcal{C}[\![cond]\!](m) \in \mathcal{P}(\mathbb{B})$,$cond \in Cond$,$m \in Mem$, $x \in \mathbb{V}$, $op \in \{+,-,\times,/\}$

$$\begin{array}{lcl}
tt \in \mathcal{C}[\![e_1 \ relop \ e_2]\!](m) & \stackrel{def}{=} & \exists v_1, v2. \begin{pmatrix} v_1 \in \mathcal{E}[\![e_1]\!](m) \\ v_2 \in \mathcal{E}[\![\ e_2]\!](m) \\ v_1 \ relop \ v2 \end{pmatrix} \\
ff \in \mathcal{C}[\![e_1 \ relop \ e_2]\!](m) & \stackrel{def}{=} & \exists v_1, v2. \begin{pmatrix} v_1 \in \mathcal{E}[\![e_1]\!](m) \\ v_2 \in \mathcal{E}[\![\ e_2]\!](m) \\ \neg(v_1 \ relop \ v2) \end{pmatrix} \\
be_1 \ \wedge \ be_2 \in \mathcal{C}[\![be_1 \ \mathbf{and} \ be_2]\!](m) & \stackrel{def}{=} & and \begin{pmatrix} be_1 \in \mathcal{C}[\![be_1]\!](m) \\ be_2 \in \mathcal{C}[\![be_2]\!](m) \end{pmatrix}
\end{array}$$

# Generating Control Flowchart Graph from Program

- A *control flow graph* is generated from the program under consideration namely $P$.
- A control flow graph $\mathcal{CFG}[\![P]\!]$ is defined by nodes $(l \in \mathcal{C})$ which are program control points of $P$, $\mathcal{C}ontrol[\![P]\!]$ and by labelled edges with actions ($\mathcal{A}ctions[\![P]\!]$) defined by the following rules :

$$\begin{array}{rcl} actions & ::= & v := exp \\ & | & skip \\ & | & \textbf{assert}\ be \end{array}$$

- A *control flow graph* is effectively defined by :
  - $\ell_{init} \in \mathcal{C}ontrol[\![P]\!]$ : the entry point
  - $\ell_{end} \in \mathcal{C}ontrol[\![P]\!]$ : the exit point
  - $\mathcal{E}dges[\![P]\!] \subseteq \mathcal{C}ontrol[\![P]\!] \times \mathcal{A}ctions[\![P]\!] \times \mathcal{C}ontrol[\![P]\!]$
- $\mathcal{CFG}[\![P]\!] = (\ell_{init}, \mathcal{E}dges[\![P]\!], \ell_{end})$

- $Mem \stackrel{def}{=} \mathbb{V} \longrightarrow \mathbb{Z}$
- Semantics of actions : $\xrightarrow{a} \subseteq Mem \times Mem$
  $m \xrightarrow{x:=e} m[x \mapsto v]$ **if** *there is a value* $v \in \mathcal{E}[\![e]\!](m)$
  $m \xrightarrow{skip} m$
  $m \xrightarrow{\textbf{assert}\ be} m]$ $if\ tt \in \mathcal{C}[\![be]\!](m)$
- Semantics for $\mathcal{CFG}[\![P]\!]$ : $\xrightarrow{P} \subseteq States \times States$
  - **If** $m \xrightarrow{a} m'$ and $(\ell_1, a, \ell_2) \in \mathcal{E}dges[\![P]\!]$, then $(\ell_1, m) \xrightarrow{P} (\ell_2, m')$
  - The set of initial states is $\{\ell_{init}\} \times Mem$
  - The set of reachable states for $P$ is denoted $\text{REACHABLE}(P)$ and defined by $[\![P]\!] = \{s | \exists s_0 \in \{\ell_{init} \times Mem : s_0 \xrightarrow[\star]{P} s\}$.

# Collecting Semantics for Programs

- Defining for each control point $\ell$ of $P$ the set of reachables values :
$$[\![P]\!]^{coll}_{\ell} = \{s | s \in States \wedge s \in [\![P]\!] \wedge \exists m \in Mem : s = (\ell, m)\}$$
- Characterizing $[\![P]\!]^{coll}_{\ell}$ : it satisfies the system of equations
$$\forall \ell \in \mathcal{C}(P). X_{\ell} = X^{init}_{\ell} \cup \bigcup_{(\ell_1, a, \ell) \in \mathcal{E}dges[\![P]\!]} [\![a]\!](X_{\ell_1}) \qquad (1)$$
- Let $a \in \mathcal{A}ctions[\![P]\!]$ and $x \subseteq Mem$.
$$[\![a]\!](x) = \{e | e \in States \wedge \exists f. f \in x \wedge f \xrightarrow{a} e\}$$
- $\forall \ell \in \mathcal{C}(P). \left( \begin{array}{l} \ell = \ell_{init} \Rightarrow X^{init}_{\ell} = Mem \\ \ell \neq \ell_{init} \Rightarrow X^{init}_{\ell} = \varnothing \end{array} \right)$

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

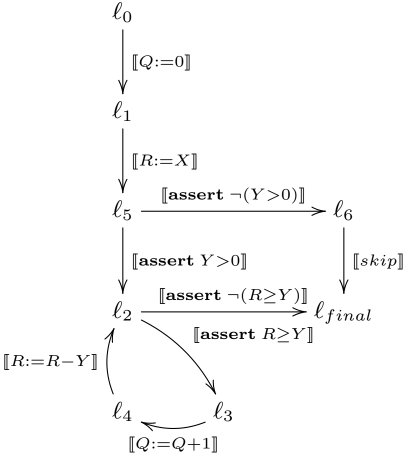
☺ Theorem Let F the function defined as follows :

- $n$ is the cardinality of $\mathcal{C}(P)$.
- $F \in \mathcal{P}(States)^n \longrightarrow \mathcal{P}(States)^n$
- If $X \in \mathcal{P}(States)^n$, then $F(X) = (\ldots, F_\ell(X), \ldots)$
- $\forall \ell \in \mathcal{C}(P).F_\ell(X) = X_\ell^{init} \cup \bigcup_{(\ell_1,a,\ell) \in \mathcal{E}dges[\![P]\!]} [\![a]\!](X_{\ell_1})$

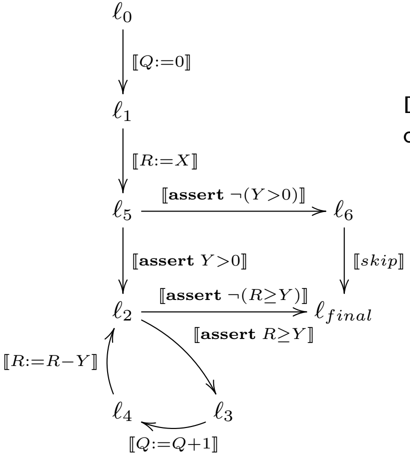
The function F is monotonic over the complete lattice $(\mathcal{P}(States)^n, \subseteq)$ and has a least fixed-point $\mu F$ defining the collecting semantics.

---

# From flowchart to equational system

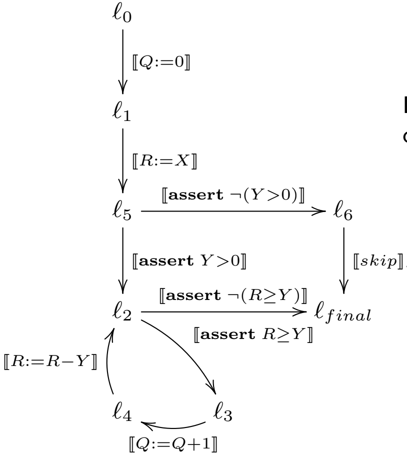


# From flowchart to equational system




Defining equational systems for the collecting semantics :

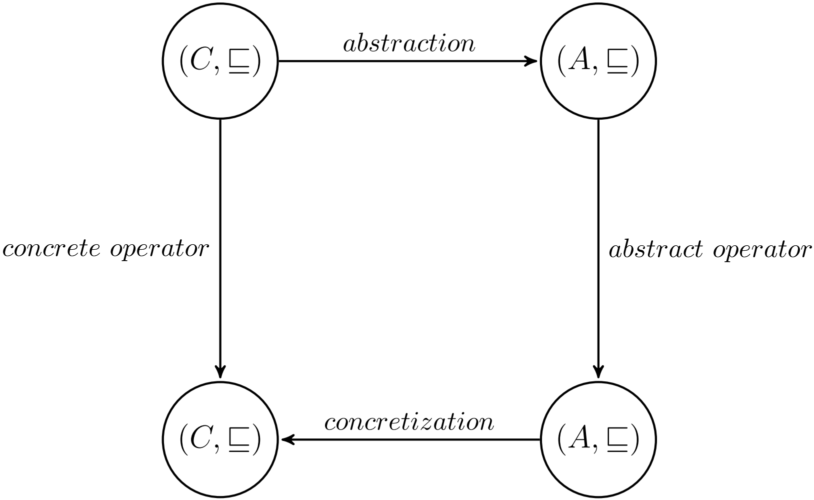
# From flowchart to equational system




Defining equational systems for the collecting semantics :

$$\begin{cases} X_0 = Mem \\ X_1 = [\![Q := 0]\!](X_0) \\ X_5 = [\![R := X]\!](X_1) \\ X_2 = [\![assert\ (Y > 0)]\!](X_5) \cup [\![R := R{-}Y]\!](X_4) \\ X_3 = [\![assert\ R \geq Y]\!](X_2) \\ X_4 = [\![Q := Q{+}1]\!](X_3) \\ X_6 = [\![assert\ \neg(Y > 0)]\!](X_5) \\ X_7 = X_6 \cup [\![assert\ \neg(R \geq Y)]\!](X_2) \end{cases}$$

# Finding Sound Abstractions for Computing



# Current Summary

# Current Subsection Summary

## Galois connections as good abstractions

- Two complete lattices $(C, \sqsubseteq_1, \sqcup_1, \sqcap_1)$ and $(A, \sqsubseteq_2, \sqcup_2, \sqcap_2)$ are supposed to be given.
- Two functions $\alpha$ and $\gamma$ are supposed to be defined as follows :
  - $\alpha \in C \longrightarrow A$
  - $\gamma \in A \longrightarrow C$
- The pair $(\alpha, \gamma)$ is a Galois connection, if it satisfies the following property : $\forall x_1 \in C, x_2 \in A. \alpha(x_1) \sqsubseteq_2 x_2 \Leftrightarrow x_1 \sqsubseteq_1 \gamma(x_2)$
- A complete lattice $A$ is a good abstraction of $L$, when there is a Galois connection between $A$ and $L$.

# Finding Sound Abstractions for Computing

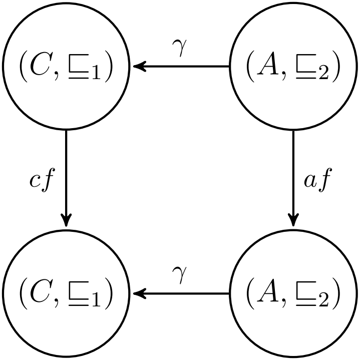
$$\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "cf"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "af"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}$$

# Finding Sound Abstractions for Computing

$$
\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "cf"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "af"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}
$$

- $a$ is a sound abstraction of $c$, if $c \sqsubseteq_1 \gamma(a)$.

## Finding Sound Abstractions for Computing




- $a$ is a sound abstraction of $c$, if $c \sqsubseteq_1 \gamma(a)$.
- functional operator : $af$ is a sound abstraction of cf, if $\forall a \in A.cf(\gamma(a)) \sqsubseteq_1 \gamma(af(a))$

## Finding Sound Abstractions for Computing

$$
\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "cf"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "af"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}
$$

- $a$ is a sound abstraction of $c$, if $c \sqsubseteq_1 \gamma(a)$.
- functional operator : $af$ is a sound abstraction of cf, if $\forall a \in A.cf(\gamma(a)) \sqsubseteq_1 \gamma(af(a))$
- relational operator : $ar$ is a sound abstraction of cr, if $\forall a \in A.cr(\gamma(a_1), \ldots, \gamma(a_n)) \sqsubseteq_1 \gamma(ac(a_1, \ldots, a_n)))$

## Galois Connections

The pair $(\alpha, \gamma)$ is a Galois connection, if it satisfies the following property : $\forall x_1 \in L, x_2 \in L.\alpha(x_1) \sqsubseteq' x_2 \Leftrightarrow x_1 \sqsubseteq \gamma(x_2)$
Notation : $L \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} L'$

## Properties of a Galois connection $L \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} L'$

- $\alpha$ and $\gamma$ are monotonic over the lattices.
- $\mathrm{id}(L) \subseteq \gamma \circ \alpha$ : $\gamma \circ \alpha$ is extensive.
- $\alpha \circ \gamma \subseteq \mathrm{id}(L')$ : $\alpha \circ \gamma$ is retractive.
- $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$
- $\alpha(x) = \bigcap' \{y \in L' | x \sqsubseteq \gamma(y)\}$
- $\gamma(y) = \bigcup \{x \in L | \alpha(x) \sqsubseteq' y\}$

### Properties

- $\gamma \circ \alpha \circ \gamma \circ \alpha = \gamma \circ \alpha$
- $\alpha \circ \gamma \circ \alpha \circ \gamma = \alpha \circ \gamma$
- We assume that $\{(\alpha_i, \gamma_i) | i \in \{1 \ldots n\}\}$ is a family of Galois connections :

  $$L_1 \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} L_2 \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \ldots L_{n-1} \underset{\alpha_{n-1}}{\overset{\gamma_{n-1}}{\leftrightarrows}} L_n$$

  Then $(\alpha_1; \ldots; \alpha_i; \ldots; \alpha_{n-1}, \gamma_{n-1}; \ldots \gamma_i; \ldots; \gamma_1)$ is a Galois connection. or equivalently $L_1 \underset{\alpha_{n-1} \circ \ldots \circ \alpha_i \circ \ldots \circ \alpha_1}{\overset{\gamma_1 \circ \ldots \gamma_i \circ \ldots \circ \gamma_{n-1}}{\leftrightarrows}}$ is a Galois connection.
- We assume that $\{(\alpha_i, \gamma_i) | i \in \{1, 2\}\}$ two Galois connections : $\alpha_1 = \alpha_2$ if, and only if, $\gamma_1 = \gamma_2$

# Current Subsection Summary

# Examples

- We consider a transition system $(S, I, t)$ where $S$ is the set of states, $I$ is the set of initial states and $t$ is a binary relation over $S$.
- A property $P$ of the transition system is a subset of $S$ : $P \subseteq S$.
- $P$ holds in $s \in S$, when $s \in P$.
- Four operators over properties can be defined as follows :
  - $\mathsf{pre}[t]P \stackrel{def}{=} \{s | s \in S \wedge \exists s'.((s,s') \in t \wedge s' \in P)\}$
  - $\widetilde{\mathsf{pre}}\,[t]P \stackrel{def}{=} \{s | s \in S \wedge \forall s'.((s,s') \in t \Rightarrow s' \in P)\}$
  - $\mathsf{post}[t]P \stackrel{def}{=} \{s | s \in S \wedge \exists s'.((s',s) \in t \wedge s' \in P)\}$
  - $\widetilde{\mathsf{post}}\,[t]P \stackrel{def}{=} \{s | s \in S \wedge \forall s'.((s',s) \in t \Rightarrow s' \in P)\}$
- Duality of operators :
  1. $\widetilde{\mathsf{pre}}\,[t]\neg P = \neg \mathsf{pre}[t]P$
  2. $\widetilde{\mathsf{post}}\,[t]\neg P = \neg \mathsf{post}[t]P$
- Galois connections over $\mathcal{P}$, the set of subsets of $S$ :

$$(\mathcal{P}, \subseteq) \underset{\mathsf{pre}[t]}{\overset{\widetilde{\mathsf{post}}[t]}{\leftrightarrows}} (\mathcal{P}, \subseteq) \qquad (\mathcal{P}, \subseteq) \underset{\mathsf{post}[t]}{\overset{\widetilde{\mathsf{pre}}[t]}{\leftrightarrows}} (\mathcal{P}, \subseteq)$$

## Examples

- Let two sets $\mathcal{L}$ standing for labels et $\mathcal{M}$ standing for memories.
- First step :
  - $\sqsubseteq$ is the partial ordering over functions using the subset relationship over function graphs : $f \sqsubseteq g$ means that $\mathbb{G}raph(f) \subseteq \mathbb{G}raph(g)$.
  - $\alpha_1 = \lambda P.\lambda l.\{m|(l,m) \in P\}$
  - $\gamma_1 = \lambda Q.\{(l,m)|l \in \mathcal{L} \wedge m \in Q(l)\}$
  - $(\mathcal{P}(\mathcal{L}\times\mathcal{M}), \subseteq) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (\mathcal{L} \longrightarrow \mathcal{P}(\mathcal{M}), \subseteq)$ is a Galois connection
- Second step :
  - Let two sets $Pred$, set of predicates, and $\mathcal{M}$, a set of memories.
  - The relationship between both sets is stating as follows : For any given predicate $p$ and any given memory $m$, $p$ holds in $m$.
  - We define $B(p) = \{m|m \in \mathcal{M} \wedge p(m)\}$, set of predicates in which $p$ holdsd.
  - Next we define :
    - $\alpha_2 = \lambda Q.\{p|p \in Pred \wedge Q \subseteq B(p)\}$
    - $\gamma_2 = \lambda P.\cap\{B(p)|p \in P\}$
  - $(\mathcal{P}(\mathcal{M}), \subseteq) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} (\mathcal{P}(Pred), \Rightarrow)$ is a Galois connection.

## Examples

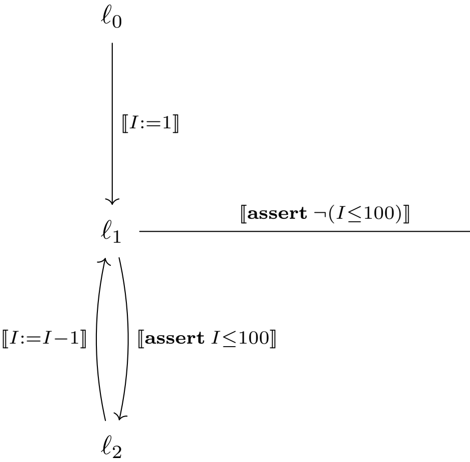
- Third step
  - $\alpha_3 = \lambda \ell.\alpha_2(Q_\ell) : Q \subseteq_1 Q' \stackrel{def}{=} \forall \ell \in \mathcal{L}.Q_\ell \subseteq Q'_\ell$.
  - $\gamma_3 = \lambda \ell.\gamma_2(P\ell) : P \Rightarrow_1 P' \stackrel{def}{=} \forall \ell \in \mathcal{L}.P_\ell \Rightarrow P'_\ell$.
  - $(\mathcal{L} \longrightarrow \mathcal{P}(\mathcal{M}), \subseteq_1) \underset{\alpha_3}{\overset{\gamma_3}{\leftrightarrows}} (\mathcal{L} \longrightarrow \mathcal{P}(Pred), \Rightarrow_1)$ is a Galois connection.

# Current Summary

# Current Subsection Summary

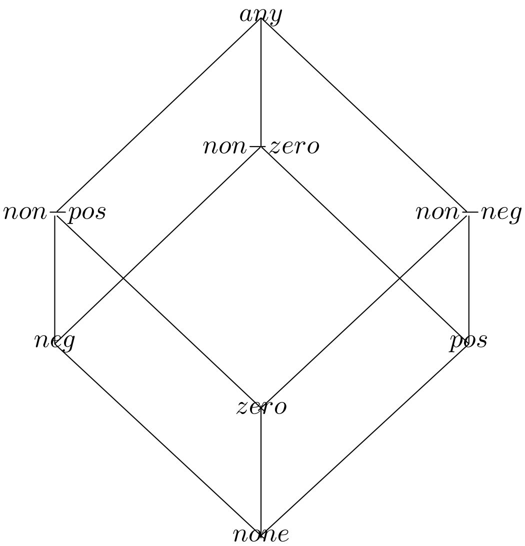
# Examples of Abstractions

$\ell_0[I := 1];$
**while** $\ell_1[I \leq 100]$ **do**
  $\ell_2[I := I+1];$
**end while**
$\ell_{final}[skip]$



# Domain of Sign

s




- Defining an abstraction for integers $\alpha \in \mathcal{P}(\mathbb{Z}) \longrightarrow \mathbb{S}igns$
- $\begin{cases} z & \alpha(z) \\ z < 0 & neg \\ z > 0 & pos \\ z = 0 & zero \end{cases}$

# Composing Galois Connections

- Abstraction by projection :
  $(\mathcal{P}(Var \longrightarrow \mathbb{Z}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (Var \longrightarrow \mathcal{P}(\mathbb{Z}), \subseteq)$
- Abstraction of signs
  $(Var \longrightarrow \mathcal{P}(\mathbb{Z}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (Var \longrightarrow \mathbb{S}igns), \subseteq)$
- Composition of abstractions :
  $(\mathcal{P}(Var \longrightarrow \mathbb{Z}), \subseteq) \underset{\alpha_{sign} \circ \alpha_\pi}{\overset{\gamma_\pi \circ \gamma_{sign}}{\leftrightarrows}} (Var \longrightarrow \mathbb{S}igns), \subseteq)$
- $\alpha = \alpha_{sign} \circ \alpha_\pi$ and $\gamma = \gamma_\pi \circ \gamma_{sign}$

# Best approximation of a function

- $L$ is the concrete domain and $L'$ is the abstract model :

$$\begin{tikzcd} L \arrow[r, "f"] & L \arrow[d, "\alpha"] \\ L' \arrow[u, "\gamma"] \arrow[r, "f'"] & L' \end{tikzcd}$$

$$f' = \alpha \circ f \circ \gamma \tag{2}$$

$f'$ is the best approximation of $f$

## Best evaluation and abstract evaluation

- Concrete states : $cv \in Var \longrightarrow \mathcal{P}(\mathbb{Z})$ : if $X$ is in $Var$, then $cv(X) \in \mathcal{P}(\mathbb{Z})$.
- Abstract states : $av \in Var \longrightarrow \mathbb{S}igns$ : if $X$ is in $Var$, then $av(X) \in \mathbb{S}igns$.
- $(\alpha, \gamma)$ is extended as : $(\alpha_1, \gamma_1)$ entre $(Var \longrightarrow \mathcal{P}(\mathbb{Z}), \subseteq)$ et $(Var \longrightarrow \mathbb{S}igns, \sqsubseteq)$. En particulier, $\alpha_1(cv) = av$ et, pour tout $X$ de $Var$, $av(X) = \alpha(cv(X))$ ; $\gamma_1(av) = cv$ et, pour tout $X$ de $Var$, $cv(X) = \gamma(av(X))$.
- Any expression $e$ can be evaluated on each domain :
  - concrete domain : $States = Var \longrightarrow \mathcal{P}(\mathbb{Z})$ : $[\![e]\!] \in (Var \longrightarrow \mathcal{P}(\mathbb{Z})) \longrightarrow \mathcal{P}(\mathbb{Z})$ and $[\![e]\!](cv)$
  - abstract domain : $AStates = Var \longrightarrow \mathbb{S}igns$ : $[\![e]\!]_a \in (Var \longrightarrow \mathbb{S}igns) \longrightarrow \mathbb{S}igns$ and $[\![e]\!]_a(av)$.

- The best abstraction is simply dedined as follows :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$.
- Applying the best approximation for assignment :
  $[\![x := e]\!]_{best}(av) = \begin{cases} av(y), y \neq x \\ [\![e]\!]_{best}(av) \end{cases}$
- $(\mathcal{P}(Var \longrightarrow \mathbb{Z}), \subseteq)$ :
  $A, B \in \mathcal{P}(\mathbb{Z}) : A+B = \{a+b | a \in A \wedge b \in B\}$
- $(Var \longrightarrow \mathbb{S}igns), \subseteq)$ :
  $x, y \in \mathbb{S}igns : x \oplus y = \alpha(\gamma(x)+\gamma(y))$
- examples :
  - $pos \oplus neg = \alpha(\gamma(pos)+\gamma(neg)) = \alpha((1,+\infty)+(-\infty,-1)) = \alpha((-\infty,+\infty)) = any$
  - $pos \oplus zero = \alpha(\gamma(pos)+\gamma(zero)) = \alpha((1,+\infty)+(0)) = \alpha((1,+\infty)) = pos$
  - Building a table for the abstract operation $\oplus$.

# Forward analysis in the domain of signs : first way

- Applying the analysis on the example

$\ell_0[X := 1];$
$\ell_1[Y := 5];$
$\ell_2[X := X{+}1];$
$\ell_3[Y := Y{-}1];$
$\ell_4[X := Y{+}X];$
$\ell_{final}[skip];$

| $\ell$ | $X$ | $Y$ |
|---|---|---|
| $\ell_0$ | $any$ | $any$ |
| $\ell_1$ | $pos$ | $any$ |
| $\ell_2$ | $pos$ | $pos$ |
| $\ell_3$ | $pos$ | $pos$ |
| $\ell_4$ | $pos$ | $non{-}neg$ |
| $\ell_{final}$ | $non{-}neg$ | $non{-}neg$ |

- $\ell_3$ to $\ell_4$ : abstract value of $Y$ is $pos$ and by $\gamma$, we obtain $(1, +\infty)$ a,d now we can compute in concrete domain $\mathbb{Z}$ $(1, +\infty)+(-1) = (0, +\infty)$. By reapplying $\alpha$ we obtain $non{-}neg$.
- Computations may be not computable and one should use techniques for accelarating the convergence like widening.
- Computing is still costly : computing now in the abstraction and defining a sound approximation of $f$.

## Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$

## Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex

## Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex
- Idea : approximation of the *best* approximation :

## Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation : $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex
- Idea : approximation of the *best* approximation :$[\![e]\!]_a$ and, for any $av$ abstract state, $[\![e]\!]_{best}(av) \sqsubseteq [\![e]\!]_a(av)$.

## Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation : $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex
- Idea : approximation of the *best* approximation :$[\![e]\!]_a$ and, for any $av$ abstract state, $[\![e]\!]_{best}(av) \sqsubseteq [\![e]\!]_a(av)$.
- Abstract semantics is defined as follows : $av \in Var \longrightarrow \mathbb{S}igns$ :

# Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex
- Idea : approximation of the *best* approximation :$[\![e]\!]_a$ and, for any $av$ abstract state, $[\![e]\!]_{best}(av) \sqsubseteq [\![e]\!]_a(av)$.
- Abstract semantics is defined as follows :
  $av \in Var \longrightarrow \mathbb{S}igns$ :
  - $[\![const]\!]_a(av) = \alpha(\{c\})$
  - $[\![x]\!]_a(av) = av(x)$
  - $[\![e_1 + e_2]\!]_a(av) = [\![e_1]\!]_a(av) \oplus [\![e_2]\!]_a(av)$
  - $[\![e_1 + e_2]\!]_a(av) = [\![e_1]\!]_a(av) \otimes [\![e_2]\!]_a(av)$

# Forward analysis in the domain of signs : second way

- Evaluation is using the *best* approximation :
  $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$
- Computing over the concrete domain is remaining complex
- Idea : approximation of the *best* approximation :$[\![e]\!]_a$ and, for any $av$ abstract state, $[\![e]\!]_{best}(av) \sqsubseteq [\![e]\!]_a(av)$.
- Abstract semantics is defined as follows :
  $av \in Var \longrightarrow \mathbb{S}igns$ :
  - $[\![const]\!]_a(av) = \alpha(\{c\})$
  - $[\![x]\!]_a(av) = av(x)$
  - $[\![e_1 + e_2]\!]_a(av) = [\![e_1]\!]_a(av) \oplus [\![e_2]\!]_a(av)$
  - $[\![e_1 + e_2]\!]_a(av) = [\![e_1]\!]_a(av) \otimes [\![e_2]\!]_a(av)$
- $\ell[X := E]$ : $[\![E]\!]_a$ in $av$ ou encore $[\![E]\!]_a(av)$ :
  $[\![Y + X + 6]\!]_a(av) = [\![Y]\!]_a(av) +_a [\![X]\!]_a(av) +_a [\![6]\!]_a(av)$.
  - $[\![Y - 1]\!]_a(av) = [\![Y]\!]_a(av) \oplus [\![-1]\!]_a(av)_a = pos \oplus neg = any$
  - $[\![Y - 1]\!]_{best}(av) = \alpha \circ [\![Y - 1]\!] \circ \gamma_1(av) == \alpha([\![Y - 1]\!](\gamma_1(av))) = \alpha([\![Y - 1]\!](\{Y \mapsto (1, +\infty)\}) = \alpha((1 + \infty) + (-1)) = \alpha((0, +\infty)) = non{-}neg$

# Sound approximations of $f$ with respect to a Galois connection

A sound approximation of $f$ with respect to a Galois connection $f'$ satisfies the following property :

$$\forall x \in L, y \in L'.\alpha(x) \sqsubseteq y \Rightarrow \alpha(f(x)) \sqsubseteq f'(y)$$

$$\begin{tikzcd}
L \arrow[r, "f"] & L \arrow[d, "\alpha"] \\
L' \arrow[u, "\gamma"] \arrow[r, "f'"] & L'
\end{tikzcd}$$

The four statements are equivalent

- $f'$ is a sound approximation of $f$ with respect to a Galois connection
- $\alpha \circ f \sqsubseteq' f' \circ \alpha$
- $\alpha \circ f \circ \gamma \sqsubseteq' f'$
- $f \circ \gamma \sqsubseteq' \gamma \circ f'$

## Defining an abstract semantics of expressions

- $[\![e]\!]_{best}(av) = \alpha \circ [\![e]\!] \circ \gamma_1(av)$ provide the best abstraction but is costly.
- Another solution is to define an abstract semantics for expressions : $[\![e]\!]_a$ such that for any $av$, $[\![e]\!]_{best}(av) \sqsubseteq [\![e]\!]_a(av)$.
- $av \in Var \longrightarrow \mathbb{S}igns$ :
  - $[\![const]\!]_a(v) = \alpha(\{c\})$
  - $[\![x]\!]_a(v) = v(x)$
  - $[\![e_1+e_2]\!]_a(v) = [\![e_1]\!]_a(v) \oplus [\![e_2]\!]_a(v)$
  - $[\![e_1+e_2]\!]_a(v) = [\![e_1]\!]_a(v) \otimes [\![e_2]\!]_a(v)$

- $[\![Y-1]\!]_a(av) = [\![Y]\!]_a(av) \oplus [\![-1]\!](av)_a = pos \oplus neg = may$
- $[\![Y-1]\!]_{best}(av) = \alpha_1 \circ [\![Y-1]\!] \circ \gamma_1(av) == \alpha_1([\![Y-1]\!](\gamma_1(av))) = \alpha_1([\![Y-1]\!](\{Y \mapsto (1,+\infty)\}) = \alpha_1((1+\infty)+(-1)) = \alpha_1((0,+\infty)) = non-neg$

# Current Summary

# Forward analysis in the domain of signs using the approximation

- Applying the analysis on the example

$\ell_0[X := 1];$
$\ell_1[Y := 5];$
$\ell_2[X := X+1];$
$\ell_3[Y := Y-1];$
$\ell_4[X := Y+X];$
$\ell_{final}[skip];$

| $\ell$ | $X$ | $Y$ |
|---|---|---|
| $\ell_0$ | $any$ | $any$ |
| $\ell_1$ | $pos$ | $any$ |
| $\ell_2$ | $pos$ | $pos$ |
| $\ell_3$ | $pos$ | $pos$ |
| $\ell_4$ | $pos$ | $any$ |
| $\ell_{final}$ | $any$ | $any$ |

- The new analysis is less precise but more efficient since we compute in the domain of signs.

# Current Subsection Summary

# Abstract Domain of Intervals

- $\mathbb{I}(\mathbb{Z}) = \{\perp\} \cup \{[l,u] | l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{\infty\}, l \leq u\}$
- $[l_1, u_1] \sqsubseteq [l_2, u_2]$ si, et seulement si, $l2 \leq l1$ et $u_1 \leq u_2$.
- $(\mathbb{I}(\mathbb{Z}), \sqsubseteq)$ est une structure partiellement ordonnée.
- 1. $[l_1, u_1] \sqcup [l_2, u_2] = [min(l_1, l_2), max(u_1, u_2)]$
  2. $[l_1, u_1] \sqcap [l_2, u_2] = \begin{cases} [max(l_1, l_2), min(u_1, u_2)] \\ \perp, si\ max(l_1, l_2) > min(u_1, u_2) \end{cases}$
- $(\mathbb{I}(\mathbb{Z}), \sqcup)$ is a complete lattice.
- 1. $\alpha(X) = \begin{cases} [min(X), max(X)] \\ \perp, si\ X = \varnothing \end{cases}$
  2. $\gamma([l, u]) = [l..u]$ et $\gamma(\perp] = \varnothing$
- $(\alpha, \gamma)$ is a Galois connexion.
- 1. $i_1 \oplus i_2 = [l_1 + l_2, u_1 + u_2]$
  2. $i_1 \ominus i_2 = [l_1 - u_2, u_1 - l_2]$
  3. $i_1 \otimes i_2 = [min(l_1 \cdot l_2, l_1 \cdot u_2, u_1 \cdot l_2, u_1 \cdot u_2, max(l_1 \cdot l_2, l_1 \cdot u_2, u_1 \cdot l_2, u_1 \cdot u_2)]$
  4. $i_1 \oslash i_2 =$ $[min(l_1/l_2, l_1/u_2, u_1/l_2, u_1/u_2, max(l_1/l_2, l_1/u_2, u_1/l_2, u_1/u_2)]$

# Current Summary

## Definition of a sound approximation of a function $f$

A function $g \in A \longrightarrow A$ is a sound approximation of a function $f \in C \longrightarrow C$, if it satisfies the following condition :
$\forall c \in C : \forall a \in A : \alpha(c) \sqsubseteq a \Rightarrow \alpha(f(c)) \sqsubseteq g(a)$

## Properties

Suppose that $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ is a Galois connection.
The four statements are equivalent

1. $g$ is a sound approximation of $f$ with respect to a Galois connection
2. $\alpha \circ f \sqsubseteq g \circ \alpha$
3. $\alpha \circ f \circ \gamma \sqsubseteq g$
4. $f \circ \gamma \sqsubseteq \gamma \circ g$
5. $f \sqsubseteq \gamma \circ g \circ \alpha$

## Example of a sound approximation of the invariant of a system

- C is the set of concrete states : $cv \in Var \longrightarrow \mathcal{P}(\mathbb{Z})$ : if if $X$ is in $Var$, then $cv(X) \in \mathcal{P}(\mathbb{Z})$.
- A is the set of abstract states : $av \in Var \longrightarrow \mathbb{S}igns$ : if $X$ is in $Var$, then $av(X) \in \mathbb{S}igns$.
- $(\alpha, \gamma)$ is extended as : $(\alpha_1, \gamma_1)$ entre $(Var \longrightarrow \mathcal{P}(\mathbb{Z}), \subseteq)$ et $(Var \longrightarrow \mathbb{S}igns, \sqsubseteq)$. En particulier, $\alpha_1(cv) = av$ et, pour tout $X$ de $Var$, $av(X) = \alpha(cv(X))$ ; $\gamma_1(av) = cv$ et, pour tout $X$ de $Var$, $cv(X) = \gamma(av(X))$.

### Computing the set of computing states of a transition system TS

- $Init \subseteq C$ is the set of initial states.
- $\textsc{Next}$ defines the transition over concrete states
- $\textsc{reachable}(TS) = \{u | u \in C \wedge (\exists x_0 . x_0 \in C \wedge (x_0 \in Init) \wedge \textsc{Next}^\star(x_0, x))\}$
- pour tout partie $U$ de $\Sigma$, $U = \mathrm{FP}(U)$
- pour tout partie $U$ de $\Sigma$, $\mathrm{FP}(U) = Init_{\mathrm{S}} \cup \longrightarrow_{[U]}$

# Sound approximation of fixed-point

$$\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "f"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "g"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}$$

$$\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "f"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "g"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}$$

**First Theorem**

- Suppose that $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ is a Galois connection
- Two functions $f \in C \rightarrow C$ and $g \in A \rightarrow A$ :
- $f$ and $g$ are monotone
- $\alpha \circ f = g \circ \alpha$.

Then $\alpha(\mu.f) = \mu.g$.

# Proof

1. $\mu g \sqsubseteq \alpha(\mu f)$
   - $f(\mu f) = \mu f$ (fixed-point property)
   - $\alpha(f(\mu f)) = \alpha(\mu f)$ (applying the relation over f and g)
   - $\alpha(f(\mu f)) = g(\alpha(\mu f)) = \alpha(\mu f)$
   - $\alpha(\mu f)$ is a fixed-point of $g$ and $\mu g \sqsubseteq \alpha(\mu f)$
2. $\alpha(\mu f) \sqsubseteq \mu g$
   - Consider $y$ a fixed-point of $g$ : $g(y) = y$ and $\mu g \sqsubseteq y$.
   - $\gamma(y)$ is a fixed-point of $f$
   - $\mu f \sqsubseteq \gamma(y)$
   - $\alpha(\mu f) \sqsubseteq y$
   - $\alpha(\mu f) \sqsubseteq \mu g$

# Sound approximation of fixed-point

$$
\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "f"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "g"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}
$$

## Sound approximation of fixed-point

$$
\begin{tikzcd}
(C, \sqsubseteq_1) \arrow[d, "f"'] & (A, \sqsubseteq_2) \arrow[l, "\gamma"'] \arrow[d, "g"] \\
(C, \sqsubseteq_1) & (A, \sqsubseteq_2) \arrow[l, "\gamma"']
\end{tikzcd}
$$

### Second Theorem

- Suppose that $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ is a Galois connection
- Two functions $f \in C \rightarrow C$ and $g \in A \rightarrow A$ :
- $f$ and $g$ are monotone
- $\alpha \circ f \sqsubseteq g \circ \alpha$.

Then $\alpha(\mu f) \sqsubseteq \mu g$.

# Example of computation

- $\alpha \in \mathcal{P}(\mathbb{Z}) \rightarrow \mathbb{S}igns : \begin{cases} z & \alpha(z) \\ z < 0 & neg \\ z > 0 & pos \\ z = 0 & zero \end{cases}$
- $f \in \mathcal{P}(\mathbb{Z}) \rightarrow \mathcal{P}(\mathbb{Z})$ where $f(X) = \{0\} \cup \{x+2 | x \in \mathbb{Z} \wedge x \in X\}$
- $g = \alpha \circ f \circ \gamma$
- $f^0 = \varnothing,\ f^1 = \{0\},\ f^2 = \{0, 2\},\ \ldots$
- $g(\perp) = \perp,\ g^1 = \alpha \circ f \circ \gamma(\perp) = [0, \infty[,\ g^2 = [0, \infty[,\ \ldots$ and $\forall i \geq 2 : g^i = [0, \infty[$.
- $\mu.g = [0, \infty[$

# Current Summary

**Definition**

$\bigtriangledown$ is a widening operator over $(L, \sqsubseteq)$ $(\bigtriangledown \in L \times L \rightarrow L)$

- For any $x$ and $y$ in $L$ : $x \sqcup y \sqsubseteq x \bigtriangledown y$
- For any sequence $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq x_3 \ldots \sqsubseteq x_i \sqsubseteq x_{i+1} \ldots$, the sequence $\{y_i | i \in \mathbb{N}\}$
  - $y_0 = x_0$
  - $y_{i+1} = y_i \bigtriangledown x_{i+1}$

  stabilizes after a finite amount of time.

## Widening

**Theorem**

If $\triangledown$ is a widening operator over $(L, \sqsubseteq)$ ($\triangledown \in L \times L \rightarrow L$), then the ascending sequence $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq x_3 \ldots \sqsubseteq x_i \sqsubseteq x_{i+1} \ldots$ defined by :

- $x_0 = \perp$
- $x_{i+1} = x_i \triangledown f(x_i)$

is eventually stationary and its limit satisfies $lfp(f) \sqsubseteq \sqsubseteq \{x_i | i \in \mathbb{N}\}$ stabilizes after a finite amount of time.

- Using $\triangledown$ instead of $\sqsubseteq$ for computing approximation of upper bound.

## Defining Widening Operator

### Intervals

- $\bot \bigtriangledown \bot = \bot$
- $\bot \bigtriangledown (l,u) = (l,u) \bigtriangledown \bot = (l,u)$
- $(l1,u1) \bigtriangledown (l2,u2) = \left( \begin{pmatrix} -\infty \ if \ l2 < l1 \\ l1 \end{pmatrix}, \begin{pmatrix} \infty \ if \ u2 > u1 \\ u1 \end{pmatrix} \right)$

## Examples of widening

- $\mathbb{I}(\mathbb{Z}) = \{\bot\} \cup \{[l,u] | l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{\infty\}, l \leq u\}$
- $(\mathbb{I}(\mathbb{Z}), \sqsubseteq)$ est une structure partiellement ordonnée.
- $[l_1, u_1] \bigtriangledown [l_2, u_2] = [cond(l_2 < l_1, -\infty, l_1), cond(u_1 < u_2, \infty, u_1)]$
- $[2,3] \bigtriangledown [1,4] = [-\infty, \infty]$
- $[0,1] \sqsubseteq [0,3]$
- $[0,1] \bigtriangledown [0,3] = [0, \infty]$.
- $[0,3] \bigtriangledown [0,2] = [0,3]$.
- $[0,2] \bigtriangledown ([0,1] \bigtriangledown [0,2]) = [0, \infty]$
- $([0,2] \bigtriangledown [0,1]) \bigtriangledown [0,2] = [0,2]$

Let us assume that $(L, \sqsubseteq)$ is a complete lattice and $f$ is a monotonic function defined from $L$ to $L$.

### Theorem

If $\bigtriangledown \in L \times L \rightarrow L$ is a widening operator, then the sequence $\{x_i | i \in \mathbb{N}\}$ defined by

- $x_0 = \perp$
- $x_{i+1} = \begin{cases} x_i \ if \ f(x_i) \sqsubseteq x_i \\ x_i \bigtriangledown f(x_i) \end{cases}$

is eventually stationary and its limit satisfies $lfp(f) \sqsubseteq \bigcup\{x_i | i \in \mathbb{N}\}$

# Current Summary

# Current Subsection Summary

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 :$
**IF** $x < y$ **THEN**
  $\ell_1 :$
  $z := y;$
  $\ell_2 :$
**ELSE**
  $\ell_3 :$
  $z := x;$
  $\ell_4 :$
**FI**
$\ell_5 :$

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $x < y$ **THEN**
  $\ell_1 :$
  $z := y;$
  $\ell_2 :$
**ELSE**
  $\ell_3 :$
  $z := x;$
  $\ell_4 :$
**FI**
$\ell_5 :$

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $x < y$ **THEN**
  $\ell_1 : y < 0 \quad x < 0$
  $z := y;$
  $\ell_2 :$
**ELSE**
  $\ell_3 :$
  $z := x;$
  $\ell_4 :$
**FI**
$\ell_5 :$

## Simple example

$\ell_0 :$

$y := -11;$

$\ell_0 : y < 0$

**IF** $x < y$ **THEN**

$\quad \ell_1 : y < 0 \quad x < 0$

$\quad z := y;$

$\quad \ell_2 : y < 0 \quad x < 0 \quad z < 0$

**ELSE**

$\quad \ell_3 :$

$\quad z := x;$

$\quad \ell_4 :$

**FI**

$\ell_5 :$

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $x < y$ **THEN**
$\ell_1 : y < 0 \quad x < 0$
$z := y;$
$\ell_2 : y < 0 \quad x < 0 \quad z < 0$
**ELSE**
$\ell_3 : y < 0 \quad x \in \mathbb{Z}$
$z := x;$
$\ell_4 :$
**FI**
$\ell_5 :$

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $x < y$ **THEN**
$\ell_1 : y < 0 \quad x < 0$
$z := x;$
$\ell_2 : y < 0 \quad x < 0 \quad z < 0$
**ELSE**
$\ell_3 : y < 0 \quad x \in \mathbb{Z}$
$z := y;$
$\ell_4 : y < 0 \quad x \in \mathbb{Z} \quad z < 0$
**FI**
$\ell_5 :$

## Simple example

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $\quad x < y \quad$ **THEN**
$\quad \ell_1 : y < 0 \quad x < 0$
$\quad z := x;$
$\quad \ell_2 : y < 0 \quad x < 0 \quad z < 0$
**ELSE**
$\quad \ell_3 : y < 0 \quad x \in \mathbb{Z}$
$\quad z := y;$
$\quad \ell_4 : y < 0 \quad x \in \mathbb{Z} \quad z < 0$
**FI**
$\ell_5 : y < 0 \quad x \in \mathbb{Z} \quad z < 0$

$\ell_0 :$
$y := -11;$
$\ell_0 : y < 0$
**IF** $x < y$ **THEN**
  $\ell_1 : y < 0 \quad x < 0$
  $z := x;$
  $\ell_2 : y < 0 \quad x < 0 \quad z < 0$
**ELSE**
  $\ell_3 : y < 0 \quad x \in \mathbb{Z}$
  $z := y;$
  $\ell_4 : y < 0 \quad x \in \mathbb{Z} \quad z < 0$
**FI**
$\ell_5 : y < 0 \quad x \in \mathbb{Z} \quad z < 0$

**Result :** $y < 0 \quad x \in \mathbb{Z} \quad z < 0$ means that $z < 0$ is an information resulting from the analysis over abstract domain of signs.

# Current Subsection Summary

# Verification by computing set of reachable states

- $\mathcal{MS}$ is $(Th(s,c), x, \text{VALS}, \text{INIT}(x), \{r_0, \ldots, r_n\})$
- $\text{NEXT} \stackrel{def}{=} r_0 \vee \ldots \vee r_n$.
- $S$ is a safety property, when
  $\forall y, x \in \text{VALS}.Init(y) \wedge \text{NEXT}^{\star}(y,x) \Rightarrow x \in S$.
- $(\mathcal{P}(\text{VALS}), \subseteq, \varnothing, \cup, \cap)$ is a complete lattice.
- $\mu F$ is defined as follows :
  - $F^0 = \varnothing$
  - $F^{i+1} = F(F_i)$, $\forall i \in \mathbb{N}$
  - $\mu F = Sup\{F^i | i \in \mathbb{N}\}$
  - For any safety property $S$, $\mu F \subseteq S$.

# Computing the least fixed-point over a finite lattice

**INPUT** $tf \in T \longrightarrow T$
**OUTPUT** $result = \mu.f$
**VARIABLES** $x, y \in T, i \in \mathbb{N}$
$\ell_0 : \{x, y \in T\}$
$x := \bot;$
$y := \bot;$
$i := 0;$
$\ell_{11} : \{x, y \in T \wedge x = F^i \wedge y = \bigcup_{k=0;k=i} F^k \wedge i \leq Card(T) \wedge i = 0\};$
**WHILE** $i \leq Card(T)$
  $\ell_1 : \{x, y \in T \wedge x = F^i \wedge y = \bigcup_{k=0;k=i} F^k \wedge i \leq Card(T)\};$
  $x := f(x);$
  $\ell_2 : \{x, y \in T \wedge x = F^{i+1} \wedge y = \bigcup_{k=0;k=i} F^k \wedge i \leq Card(T)\};$
  $y := x \sqcup y;$
  $\ell_3 : \{x, y \in T \wedge x = F^{i+1} \wedge y = \bigcup_{k=0;k=i+1} F^k \wedge i \leq Card(T)\};$
  $i := i+1;$
  $\ell_4 : \{x, y \in T \wedge x = F^i \wedge y = \bigcup_{k=0;k=i} F^k \wedge i \leq Card(T)+1\};$
**OD**;
$\ell_5 : \{x, y \in T \wedge x = F^i \wedge y = \bigcup_{k=0;k=i} F^k \wedge i = Card(T)+1\};$
$result := y;$
$\ell_6 : \{x, y \in T \wedge x = F^i \wedge y = \bigcup_{k=0;k=i} F^k \wedge i = Card(T)+1 \wedge result = y\};$

## Conclusion and Comments

- Abstract interpretation is a general framework for defining sound approximation of the semantics of computer programs, based on monotonic functions over ordered sets, especially lattices.
- Main concrete application is formal static analysis, the automatic extraction of information about the possible executions of computer programs.
- When defining an abstract domain, it can be finite (diomain of signs) or infinite (domain of intervals) : it means that we have to manage undecidability questions for computing fixed-points.
- interproc is a tool that can be used for analysing recursive programs and for playing with abstract interpretation.

# Current Summary

# Tryptique $(\mathcal{D}, \mathcal{S}, \mathcal{R})$

- $\mathcal{R}$ : exigences du système.

- $\mathcal{R}$ : exigences du système.
- $\mathcal{D}$ : domaine du problème.

## Tryptique $(\mathcal{D}, \mathcal{S}, \mathcal{R})$

- $\mathcal{R}$ : exigences du système.
- $\mathcal{D}$ : domaine du problème.
- $\mathcal{S}$ : système répondant aux spécifications.

- $\mathcal{R}$ : exigences du système.
- $\mathcal{D}$ : domaine du problème.
- $\mathcal{S}$ : système répondant aux spécifications.

$$\mathcal{D}, \mathcal{S} \quad \text{SATISFAIT} \quad \mathcal{R}$$

- $\mathcal{R}$ : exigences du système.
- $\mathcal{D}$ : domaine du problème.
- $\mathcal{S}$ : système répondant aux spécifications.

$$\mathcal{D}, \mathcal{S} \quad \text{SATISFAIT} \quad \mathcal{R}$$

- $\mathcal{R}$ : pre/post.
- $\mathcal{D}$ : entiers, réels, ...
- $\mathcal{S}$ : code, procédure, programme, ...

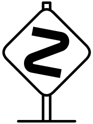
# Sommaire des résultats

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \textbf{pre}(\textsc{Alg})(v) \\ \textbf{post}(\textsc{Alg})(v_0, v) \end{cases}$$

| $\mathcal{D}$ |
|---|
| $\textbf{pre}(\textsc{Alg})(v)$<br>$\textbf{post}(\textsc{Alg})(v_0, v)$ |
| $\textsc{Alg}$ |

# Sommaire des résultats

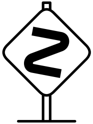
$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \textbf{pre}(\textsc{Alg})(v) \\ \textbf{post}(\textsc{Alg})(v_0, v) \end{cases}$$



**Vérification de conditions de vérification**

| $\mathcal{D}$ |
|---|
| $\textbf{pre}(\textsc{Alg})(v)$<br>$\textbf{post}(\textsc{Alg})(v_0, v)$ |
| $\textsc{Alg}$ |

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \textbf{pre}(\textsc{Alg})(v) \\ \textbf{post}(\textsc{Alg})(v_0, v) \end{cases}$$

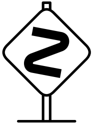


**Vérification de conditions de vérification**

| $\mathcal{D}$ |
|---|
| $\textbf{pre}(\textsc{Alg})(v)$<br>$\textbf{post}(\textsc{Alg})(v_0, v)$ |
| $\textsc{Alg}$ |

- Vérification des conditions de vérification avec un model-checker par exploration de tous les états.

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \textbf{pre}(\textsc{Alg})(v) \\ \textbf{post}(\textsc{Alg})(v_0, v) \end{cases}$$



**Vérification de conditions de vérification**

$$\begin{array}{|l|} \hline \mathcal{D} \\ \hline \textbf{pre}(\textsc{Alg})(v) \\ \textbf{post}(\textsc{Alg})(v_0, v) \\ \hline \textsc{Alg} \\ \hline \end{array}$$

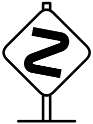
- Vérification des conditions de vérification avec un model-checker par exploration de tous les états.
- Vérification des conditions de vérification avec un outil de preuve formelle.

# Sommaire des résultats

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \texttt{requires } \textsc{Alg}(v) \\ \texttt{ensures } \textsc{Alg}(v_0, v) \end{cases}$$

| $\mathcal{D}$ |
|---|
| $\texttt{requires } \textsc{Alg}(v)$<br>$\texttt{ensures } \textsc{Alg}(v_0, v)$ |
| $\textsc{Alg}$ |

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \texttt{requires}\ \textsc{Alg}(v) \\ \texttt{ensures}\ \textsc{Alg}(v_0, v) \end{cases}$$

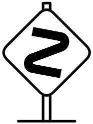


**Vérification de conditions de vérification**

| $\mathcal{D}$ |
| --- |
| $\texttt{requires}\ \textsc{Alg}(v)$<br>$\texttt{ensures}\ \textsc{Alg}(v_0, v)$ |
| $\textsc{Alg}$ |

# Sommaire des résultats

$$\mathcal{D}, \textsc{Alg} \quad \textsc{satisfait} \quad \begin{cases} \texttt{requires}\ \textsc{Alg}(v) \\ \texttt{ensures}\ \textsc{Alg}(v_0, v) \end{cases}$$



**Vérification de conditions de vérification**

| $\mathcal{D}$ |
| --- |
| $\texttt{requires}\ \textsc{Alg}(v)$<br>$\texttt{ensures}\ \textsc{Alg}(v_0, v)$ |
| $\textsc{Alg}$ |

- Vérification des conditions de vérification avec un outil de preuve formelle QED
- Vérification des conditions de vérification avec un outil de preuve formelle Alt-Ergo

# Current Summary

## Summary on techniques and tools

- Abstract Interpretation is an effective, general and scalable technique for analysing programs : invariance properties and safety properties (Astrée, Frama-c . . . ) ( **abstract domains**)
- Model Checking is an effective and limited technique for analysing programs with respect to temporal properties as invariance, safety, liveness, . . . properties (SPIN/Promela, PAT, Toolbox/TLA, . . . ).
- Proof assistants (WHY3, B, Boogie, Visual Eiffel, GNAT, . . . à)