Abstract Interpretation

Thomas Jensen

SOS, Master Recherche Science Informatique, U. Rennes

Slides by David Pichardie and Thomas Jensen

Outline

Why abstract interpretation ?

The bad news : Rice's theorem :

> *For a Turing-complete programming language, for any non-trivial property, the question of whether the computation of a given program satisfies this property is undecidable.*

Rice's theorem for static analyses :

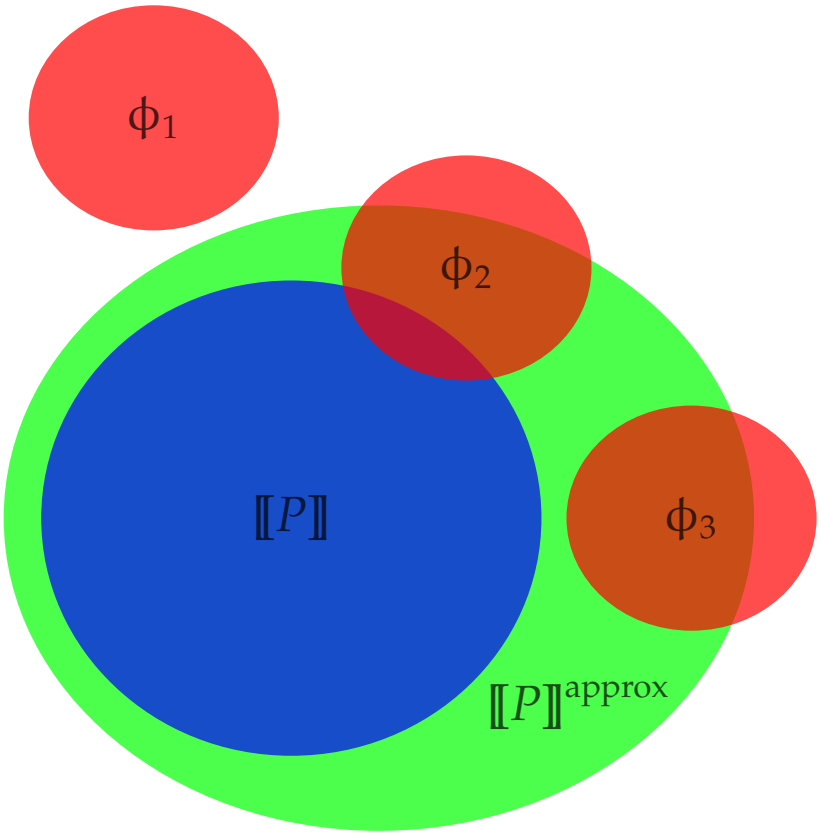
> *No fully-automated static analyser can decide a non-trivial property for all programs in a finite time.*

Solutions :

- verify the program interactively with the help of the user (deductive methods)
- computes only an approximation of the behavior of the program.

Static program analysis is about computing "good" approximations.

A static analysis computes an approximation



- P is safe w.r.t. ϕ_1 and the analyser proves it

$$[\![P]\!] \cap \phi_1 = \emptyset \qquad [\![P]\!]^{\text{approx}} \cap \phi_1 = \emptyset$$

- P is unsafe w.r.t. ϕ_2 and the analyser warns about it

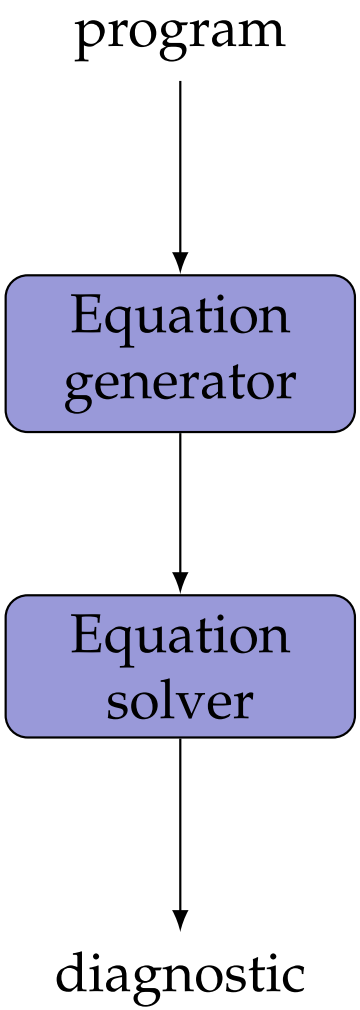
$$[\![P]\!] \cap \phi_2 \neq \emptyset \qquad [\![P]\!]^{\text{approx}} \cap \phi_2 \neq \emptyset$$

- but P is safe w.r.t. ϕ_3 and the analyser can't prove it (this is called a *false alarm*)

$$[\![P]\!] \cap \phi_3 = \emptyset \qquad [\![P]\!]^{\text{approx}} \cap \phi_3 \neq \emptyset$$

$[\![P]\!]$:	concrete semantics	(not computable)
ϕ_1, ϕ_2, ϕ_3 :	set of error states	(computable)
$[\![P]\!]^{\text{approx}}$:	analyser result (over-approximation)	(computable)

Common structure of analyses



An analysis can be separated into two parts :

1. From a program description, producing an equation system (analysis specification)
   - the solutions of the system must be proved correct w.r.t. the program semantics
2. Solving the system
   - *fixpoint* iterations in *lattice* structures

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

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

$$\begin{array}{lll} P & ::= & x\ C\ 0 \wedge y\ C\ 0 \\ C & ::= & < \mid \leqslant \mid = \mid > \mid \geqslant \end{array}$$

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

```
x = 0; y = 0;
```
$x \geqslant 0 \wedge y \geqslant 0$
```
while (x<6) {
  if (?) {
```
$x \geqslant 0 \wedge y \geqslant 0$
```
    y = y+2;
```
$x \geqslant 0 \wedge y > 0$
```
  };
```
$x \geqslant 0 \wedge y \geqslant 0$
```
  x = x+1;
```
$x > 0 \wedge y \geqslant 0$
```
}
```
$x > 0 \wedge y \geqslant 0$

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;
```
$\{1 \leqslant i \leqslant T.length\}$

```
while i<T.length {
```
$\{1 \leqslant i \leqslant T.length - 1\}$

```
    p = T[i]; j = i-1;
```
$\{1 \leqslant i \leqslant T.length - 1 \ \wedge \ -1 \leqslant j \leqslant i - 1\}$

```
    while 0<=j and T[j]>p {
```
$\{1 \leqslant i \leqslant T.length - 1 \ \wedge \ 0 \leqslant j \leqslant i - 1\}$

```
        T[j+1]=T[j]; j = j-1;
```
$\{1 \leqslant i \leqslant T.length - 1 \ \wedge \ -1 \leqslant j \leqslant i - 2\}$

```
    };
```
$\{1 \leqslant i \leqslant T.length - 1 \ \wedge \ -1 \leqslant j \leqslant i - 1\}$

```
    T[j+1]=p; i = i+1;
```
$\{2 \leqslant i \leqslant T.length + 1 \ \wedge \ -1 \leqslant j \leqslant i - 2\}$

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

Outline

While language : syntax

$$
\begin{array}{rll}
Exp ::= & n & n \in \mathbb{Z} \\
| & ? & \\
| & x & x \in \mathbb{V} \\
| & Exp\ o\ Exp & o \in \{+, -, \times\} \\
 & & \\
test ::= & Exp\ c\ Exp & c \in \{=, \neq, <, \leqslant\} \\
| & test\ \textbf{and}\ test & \\
| & test\ \textbf{or}\ test & \\
 & & \\
Stm ::= & {}^{l}[x := Exp] & l \in \mathbb{P} \\
| & {}^{l}[\texttt{skip}] & \\
| & \texttt{if}\ {}^{l}[test]\ \{\ Stm\ \}\ \{\ Stm\ \} & \\
| & \texttt{while}\ {}^{l}[test]\ \{\ Stm\ \} & \\
| & Stm\ ;\ Stm & \\
Prog ::= & [Stm]^{end} & end \in \mathbb{P}
\end{array}
$$

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

While syntax : example

$$
\begin{array}{l}
[^0[x :=?]; \\
\texttt{if}\ ^1[x < 0]\ \{ \\
\qquad \texttt{while}\ ^2[x < 0]\ \{ \\
\qquad\qquad {}^3[x := x + 1]; \\
\qquad \}; \\
\qquad {}^4[y := x]; \\
\}\ \texttt{else}\ \{ \\
\qquad {}^5[y := 0]; \\
\};]^6
\end{array}
$$

While language : semantics

Semantic domains

$$\begin{array}{rcl} Env & \stackrel{\text{def}}{=} & \mathbb{V} \to \mathbb{Z} \\ State & \stackrel{\text{def}}{=} & \mathbb{P} \times Env \end{array}$$

Semantics of expressions

$$\mathcal{A}[\![e]\!]\rho \in \mathcal{P}(\mathbb{Z}), \quad e \in Exp,\ \rho \in Env$$

$$\begin{array}{lcl} \mathcal{A}[\![n]\!]\rho & = & \{ n \} \\ \mathcal{A}[\![?]\!]\rho & = & \mathbb{Z} \\ \mathcal{A}[\![x]\!]\rho & = & \{ \rho(x) \},\ x \in \mathbb{V} \\ \mathcal{A}[\![e_1\ o\ e_2]\!]\rho & = & \{ v_1\, o\, v_2 \mid v_1 \in \mathcal{A}[\![e_1]\!]\rho,\ v_2 \in \mathcal{A}[\![e_2]\!]\rho \\ & & \qquad o \in \{+, -, \times\} \} \end{array}$$

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

Semantics of tests

$$\mathcal{B}[\![t]\!]\rho \in \mathcal{P}(\mathbb{B}), \quad t \in \mathit{test},\ \rho \in \mathit{Env} \quad \mathbb{B} = \{\mathbf{tt}, \mathbf{ff}\}$$

$$\frac{v_1 \in \mathcal{A}[\![e_1]\!]\rho \quad v_2 \in \mathcal{A}[\![e_2]\!]\rho \quad v_1\ \bar{c}\ v_2}{\mathbf{tt} \in \mathcal{B}[\![e_1\ c\ e_2]\!]\rho}$$

$$\frac{v_1 \in \mathcal{A}[\![e_1]\!]\rho \quad v_2 \in \mathcal{A}[\![e_2]\!]\rho \quad \neg(v_1\ \bar{c}\ v_2)}{\mathbf{ff} \in \mathcal{B}[\![e_1\ c\ e_2]\!]\rho}$$

$$\frac{b_1 \in \mathcal{B}[\![t_1]\!]\rho \quad b_2 \in \mathcal{B}[\![t_2]\!]\rho}{b_1 \wedge_{\mathbb{B}} b_2 \in \mathcal{B}[\![t_1\ \mathbf{and}\ t_2]\!]\rho}$$

$$\frac{b_1 \in \mathcal{B}[\![t_1]\!]\rho \quad b_2 \in \mathcal{B}[\![t_2]\!]\rho}{b_1 \vee_{\mathbb{B}} b_2 \in \mathcal{B}[\![t_1\ \mathbf{or}\ t_2]\!]\rho}$$

Structural Operational Semantics

Small-step semantics

$$\frac{v \in \mathcal{A}[\![a]\!]\rho}{({}^{l}[x := a], \rho) \Rightarrow \rho[x \mapsto v]} \qquad \frac{}{({}^{l}[\mathtt{skip}], \rho) \Rightarrow \rho}$$

$$\frac{(S_1, \rho) \Rightarrow \rho'}{(S_1 \,;\, S_2, \rho) \Rightarrow (S_2, \rho')} \qquad \frac{(S_1, \rho) \Rightarrow (S_1', \rho')}{(S_1 \,;\, S_2, \rho) \Rightarrow (S_1' \,;\, S_2, \rho')}$$

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

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

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

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

Reachable states

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

Formally :

$$[\![[P]^{\text{end}}]\!]_{\text{SOS}} = \left\{ (k, \rho) \;\middle|\; \begin{array}{l} \exists \rho_0 \in Env, \\ \quad \exists S \in Stm,\ (P, \rho_0) \Rightarrow^* (S, \rho) \text{ and } k = \text{entry}(S) \\ \quad \text{or } (P, \rho_0) \Rightarrow^* \rho \text{ and } k = \text{end} \end{array} \right\}$$

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*

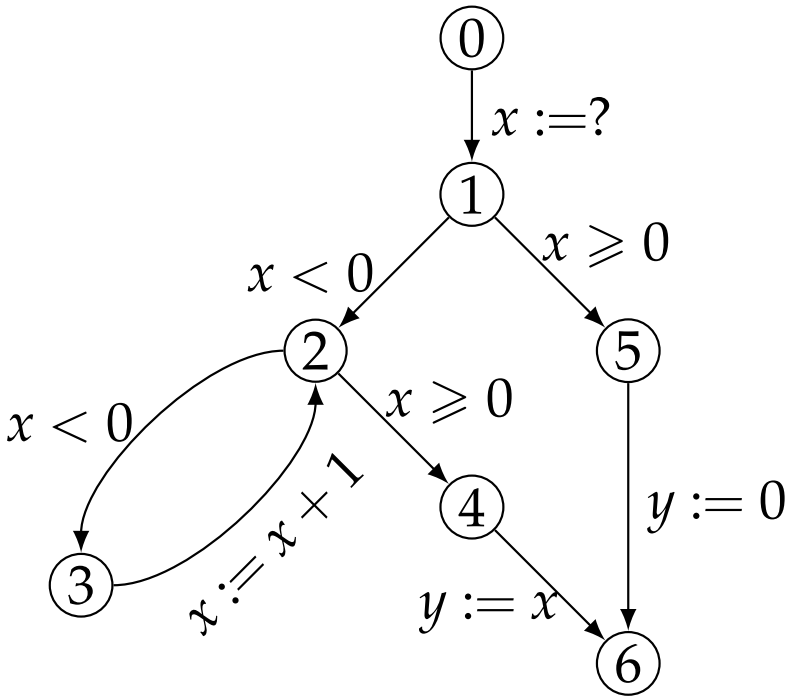
Instr ::=	x	:= *Exp*	*assignment*
|	`assume` *test*		*execution continues only if the test succeds*

- 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,
  - $S \subseteq \mathbb{P} \times \mathit{Instr} \times \mathbb{P}$ 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

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



assume is left implicit

Small-step semantics of cfg

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

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

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

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

Reachable states for control flow graphs

$$[\![(k_{\text{init}}, S, k_{\text{end}})]\!]_{\text{CFG}} = \{\ (k, \rho) \mid \exists \rho_0 \in Env,\ (k_{\text{init}}, \rho_0) \rightarrow^*_{(k_{\text{init}}, S, k_{\text{end}})} (k, \rho)\ \}$$

Control flow graph generation (1/2)

$cfg_l(S)$ computes the edges of the control flow graph of S using l as final label.

$$
\begin{aligned}
cfg_l &\in Stm \to \mathcal{P}(\mathbb{P} \times Instr \times \mathbb{P}), \quad l \in \mathbb{P} \\
cfg_{l'}\left({}^{l}[x := e]\right) &= \{(l, x := e, l')\} \\
cfg_{l'}\left({}^{l}[\texttt{skip}]\right) &= \{(l, \texttt{assume}\ T, l')\} \qquad \text{with } T \equiv 0 = 0 \\
cfg_{l'}\left(\texttt{if}\ {}^{l}[t]\ \{\ S_1\ \}\{\ S_2\ \}\right) &= \{(l, \texttt{assume}\ t, entry(S_1))\} \cup \\
&\quad \{(l, \texttt{assume}\ neg(t), entry(S_2))\} \cup cfg_{l'}(S_1) \cup cfg_{l'}(S_2) \\
cfg_{l'}\left(\texttt{while}\ {}^{l}[t]\ \{\ S\ \}\right) &= \{(l, \texttt{assume}\ t, entry(S))\} \cup \\
&\quad cfg_l(S) \cup \{(l, \texttt{assume}\ neg(t), l')\} \\
cfg_{l'}\left(S_1;\ S_2\right) &= cfg_{entry(S_2)}(S_1) \cup cfg_{l'}(S_2)
\end{aligned}
$$

$$
\begin{aligned}
cfg &\in Prog \to \mathbb{P} \times \mathcal{P}(\mathbb{P} \times Instr \times \mathbb{P}) \times \mathbb{P} \\
cfg([P]^{end}) &= (entry(P), cfg_{end}(P), end)
\end{aligned}
$$

Control flow graph generation (2/2)

Test negation :

$$
\begin{aligned}
neg(e_1 = e_2) &= e_1 \neq e_2 \\
neg(e_1 \neq e_2) &= e_1 = e_2 \\
neg(e_1 < e_2) &= e_2 \leqslant e_1 \\
neg(e_1 \leqslant e_2) &= e_2 < e_1 \\
neg(t_1 \textbf{ and } t_2) &= neg(t_1) \textbf{ or } neg(t_1) \\
neg(t_1 \textbf{ or } t_2) &= neg(t_1) \textbf{ and } neg(t_1)
\end{aligned}
$$

Correctness of *cfg*

Theorem

For all program p,

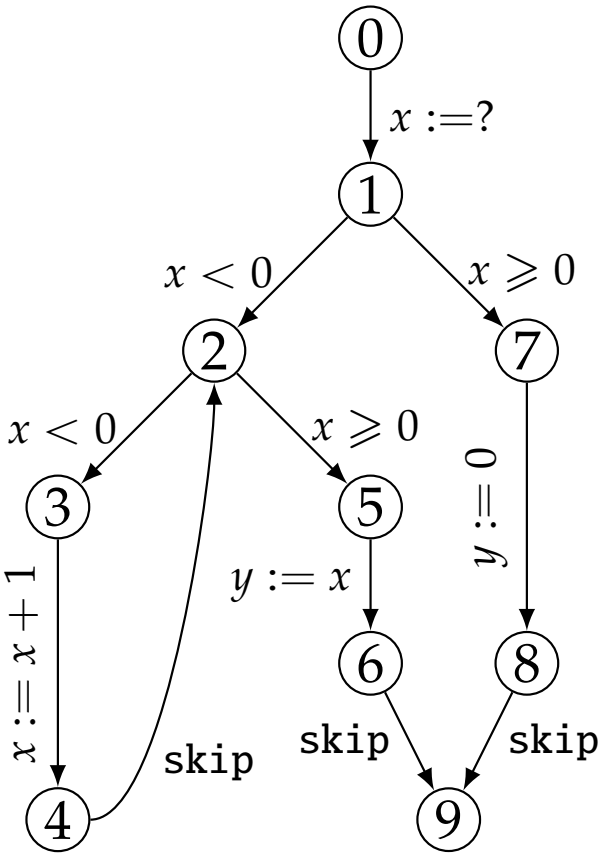
$$[\![cfg(p)]\!]_{CFG} = [\![p]\!]_{SOS}$$

From now on, we write $[\![p]\!]$ instead of $[\![cfg(p)]\!]_{\text{CFG}}$ and we identify $cfg(p)$ and p.

Outline

From CFGs to fixpoint equations

control flow graph



Abstract interpretation

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

Abstract semantics : the ingredients

- A lattice structure $(Env^\sharp, \sqsubseteq^\sharp_{Env}, \sqcup^\sharp_{Env}, \sqcap^\sharp_{Env}, \bot^\sharp_{Env}, \top^\sharp_{Env})$
  - $\sqsubseteq^\sharp_{Env}$ is an approximation of $\subseteq$
  - $\sqcup^\sharp_{Env}$ is an approximation of $\cup$
  - $\sqcap^\sharp_{Env}$ is an approximation of $\cap$
  - $\bot^\sharp_{Env}$ is an approximation of $\emptyset$
  - $\top^\sharp_{Env}$ is an approximation of Env
- For all $x \in \mathbb{V}$, $e \in Exp$, an approximation of $[\![x := e]\!]$:

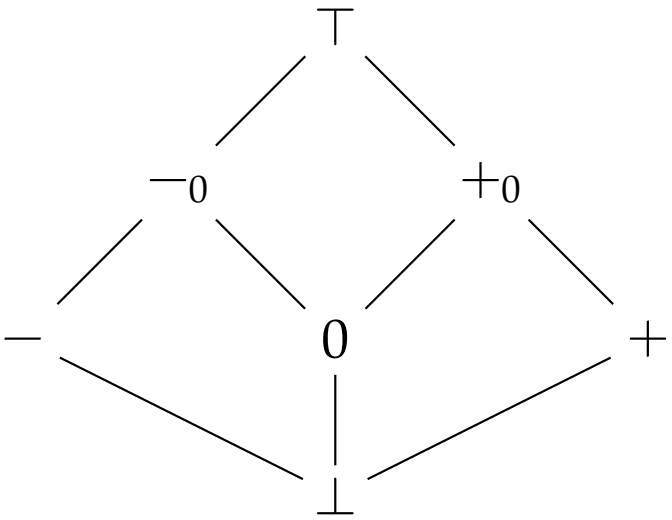
$$[\![x := e]\!]^\sharp \in Env^\sharp \to Env^\sharp$$

- For all $t \in test$, an approximation of $[\![t]\!]$:

$$[\![t]\!]^\sharp \in Env^\sharp \to Env^\sharp$$

.

An abstraction by signs



$\bot$	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	$\{ z \mid z \leqslant 0 \}$
$+_0$	represents the property	$\{ z \mid z \geqslant 0 \}$
$\top$	represents the property	$\mathbb{Z}$

$\boldsymbol{Env}^\sharp \stackrel{\text{def}}{=} \mathbb{V} \to \boldsymbol{Sign}$: a sign is associated with each variable.

An abstraction by signs : example

$$
\begin{array}{lcl}
X_0^\sharp & = & \top_{Env}^\sharp \\
X_1^\sharp & = & [\![x :=?]\!]^\sharp\,(X_0^\sharp) \\
X_2^\sharp & = & [\![x < 0]\!]^\sharp\,(X_1^\sharp) \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & [\![x < 0]\!]^\sharp\,(X_2^\sharp) \\
X_4^\sharp & = & [\![x := x + 1]\!]^\sharp\,(X_3^\sharp) \\
X_5^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_2^\sharp) \\
X_6^\sharp & = & [\![y := x]\!]^\sharp\,(X_5^\sharp) \\
X_7^\sharp & = & [\![x \geqslant 0]\!]^\sharp\,(X_1^\sharp) \\
X_8^\sharp & = & [\![y := 0]\!]^\sharp\,(X_7^\sharp) \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}
\quad \xrightarrow[\text{simplifies into}]{\text{which}} \quad
\begin{array}{lcl}
X_0^\sharp & = & [x : \top;\ y : \top] \\
X_1^\sharp & = & X_0^\sharp[x \mapsto \top] \\
X_2^\sharp & = & X_1^\sharp[x \mapsto -] \sqcup^\sharp X_4^\sharp \\
X_3^\sharp & = & X_2^\sharp[x \mapsto -] \\
X_4^\sharp & = & X_3^\sharp[x \mapsto \mathrm{succ}^\sharp(X_3^\sharp(x))] \\
X_5^\sharp & = & X_2^\sharp[x \mapsto +_0] \\
X_6^\sharp & = & X_5^\sharp[y \mapsto X_5^\sharp(x)] \\
X_7^\sharp & = & X_1^\sharp[x \mapsto +_0] \\
X_8^\sharp & = & X_7^\sharp[y \mapsto 0] \\
X_9^\sharp & = & X_6^\sharp \sqcup^\sharp X_8^\sharp
\end{array}
$$

with

$$
\begin{array}{rcl}
\mathrm{succ}^\sharp(\bot) & = & \bot \\
\mathrm{succ}^\sharp(-) & = & -_0 \\
\mathrm{succ}^\sharp(0) & = & \mathrm{succ}^\sharp(+) = \mathrm{succ}^\sharp(+_0) = + \\
\mathrm{succ}^\sharp(-_0) & = & \mathrm{succ}^\sharp(\top) = \top
\end{array}
$$

Equation solving by fixpoint iteration

Initialise x and y to the smallest abstract value $\bot$.
Recompute new approximations using the equations.
We write (v_x, v_y) for the abstract environment $[x : v_x; y : v_y]$.

Equations	0.	1.	2.	3.	4.
$X_0^\sharp = [x : \top;\ y : \top]$	$(\bot, \bot)$	$(\top, \top)$	$(\top, \top)$	$(\top, \top)$	$(\top, \top)$
$X_1^\sharp = X_0^\sharp[x \mapsto \top]$	$(\bot, \bot)$	$(\top, \bot)$	$(\top, \top)$	$(\top, \top)$	$(\top, \top)$
$X_2^\sharp = X_1^\sharp[x \mapsto -] \sqcup^\sharp X_4^\sharp$	$(\bot, \bot)$	$(-, \bot)$	$(-, \bot)$	$(-_0, \top)$	$(-_0, \top)$
$X_3^\sharp = X_2^\sharp[x \mapsto -]$	$(\bot, \bot)$	$(-, \bot)$	$(-, \bot)$	$(-, \top)$	$(-, \top)$
$X_4^\sharp = X_3^\sharp[x \mapsto \mathrm{succ}^\sharp(X_3^\sharp(x))]$	$(\bot, \bot)$	$(\bot, \bot)$	$(-_0, \bot)$	$(-_0, \bot)$	$(-_0, \top)$
$X_5^\sharp = X_2^\sharp[x \mapsto +_0]$	$(\bot, \bot)$	$(+_0, \bot)$	$(+_0, \bot)$	$(+_0, \bot)$	$(+_0, \bot)$
$X_6^\sharp = X_5^\sharp[y \mapsto X_5^\sharp(x)]$	$(\bot, \bot)$	$(\bot, \bot)$	$(+_0, +_0)$	$(+_0, +_0)$	$(+_0, +_0)$
$X_7^\sharp = X_1^\sharp[x \mapsto +_0]$	$(\bot, \bot)$	$(+_0, \bot)$	$(+_0, \bot)$	$(+_0, \top)$	$(+_0, \top)$
$X_8^\sharp = X_7^\sharp[y \mapsto 0]$	$(\bot, \bot)$	$(\bot, 0)$	$(+_0, 0)$	$(+_0, 0)$	$(+_0, 0)$
$X_9^\sharp = X_6^\sharp \sqcup^\sharp X_8^\sharp$	$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, 0)$	$(+_0, +_0)$	$(+_0, +_0)$

After the fourth iteration, the values of the $X_i^\sharp$ do not change.
We have found a fixpoint.

Exercise

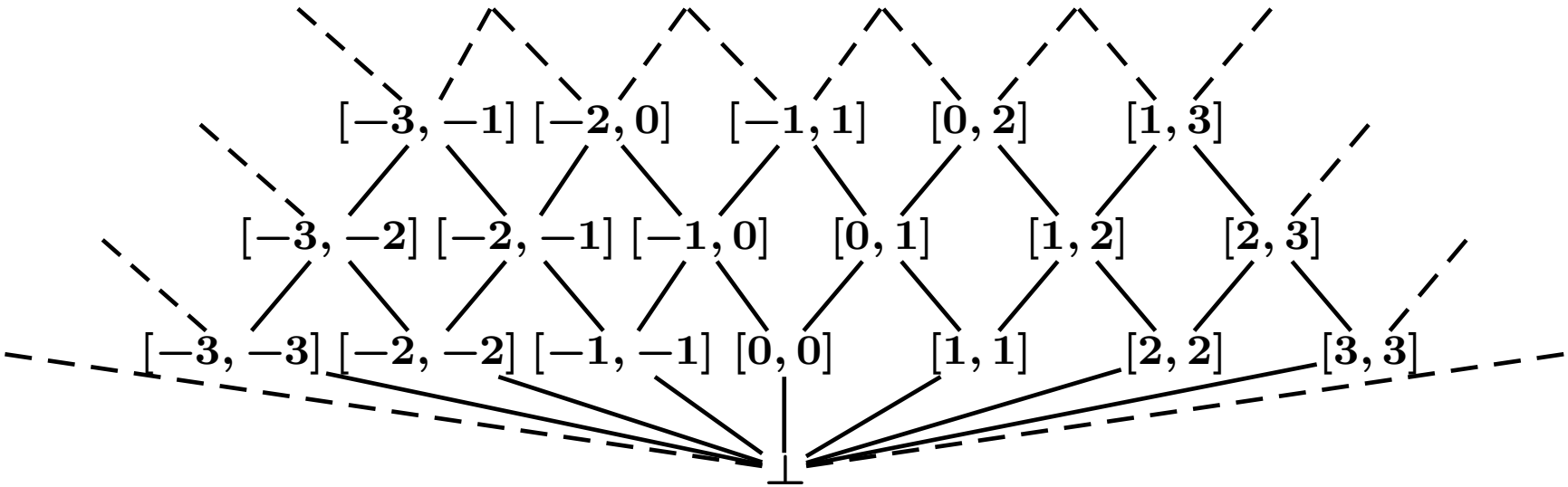
Perform a sign analysis of the program

$$
\begin{array}{l}
[^0[x := -1]; \\
\texttt{if}\ ^1[x < 0]\ \{ \\
\qquad ^2[y := -3]; \\
\}\ \texttt{else}\ \{ \\
\qquad ^3[y := 3]; \\
\};]^4
\end{array}
$$

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

Outline

The lattice of intervals



The lattice of intervals

Elements :

$$\mathrm{Itv} \stackrel{\text{def}}{=} \{\, [a,b] \mid a,b \in \overline{\mathbb{Z}},\ a \leqslant b \,\} \cup \{\bot\} \quad \text{with } \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$$

Order :

$$\frac{I \in \mathrm{Itv}}{\bot \sqsubseteq_{\mathrm{Itv}} I} \qquad \frac{c \leqslant a \quad b \leqslant d \quad a,b,c,d \in \overline{\mathbb{Z}}}{[a,b] \sqsubseteq_{\mathrm{Itv}} [c,d]}$$

Lattice operations :

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

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

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

$$\rho_{\text{Itv}}(a,b) = \begin{cases} [a,b] & \text{if } a \leqslant b, \\ \bot & \text{otherwise} \end{cases}$$

Least and greatest element :

$$\begin{array}{lcl} \bot_{\text{Itv}} & \stackrel{\text{def}}{=} & \bot \\ \top_{\text{Itv}} & \stackrel{\text{def}}{=} & [-\infty, +\infty] \end{array}$$

Abstraction and concretisation :

$$\begin{array}{lcll} \alpha_{\text{Itv}}(S) & \stackrel{\text{def}}{=} & \bot & \text{if } S = \emptyset \\ \alpha_{\text{Itv}}(S) & \stackrel{\text{def}}{=} & [\min(S), \max(S)] & \text{otherwise} \end{array}$$

$$\begin{array}{rcl} \gamma_{\text{Itv}}(\bot) & \stackrel{\text{def}}{=} & \emptyset \\ \gamma_{\text{Itv}}([a,b]) & \stackrel{\text{def}}{=} & \{\, z \in \mathbb{Z} \mid a \leqslant z \text{ and } z \leqslant b \,\} \end{array}$$

Abstraction of basic functions

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

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

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

Example :

$$
+^\sharp([2,\infty],[3,4]) = [5,\infty]
$$

Abstract comparison operators

Update knowledge about the value of variables.

Example : if $x \mapsto [2,5]$ and $y \mapsto [3,7]$ and we know that the test $x = y$ succeeds then we can refine our description of x and y :

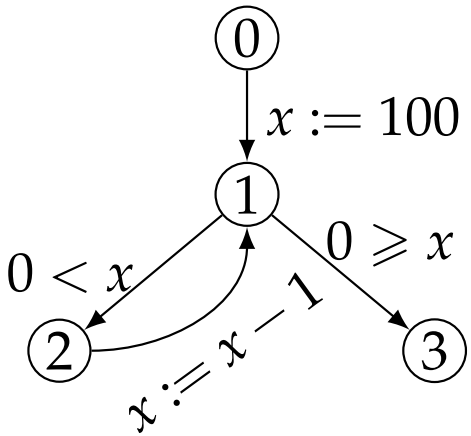
$$[\![=]\!]^\sharp_{\text{comp}}\ ([2,5],[3,7]) = ([3,5],[3,5])$$

In general :

$$\begin{aligned}
[\![=]\!]^\sharp_{\text{comp}}\ ([a,b],[c,d]) &= ([a,b] \sqcap_{\text{Itv}} [c,d], [a,b] \sqcap_{\text{Itv}} [c,d]) \\
[\![<]\!]^\sharp_{\text{comp}}\ ([a,b],[c,d]) &= ([a,b] \sqcap_{\text{Itv}} [-\infty, d-1], [a+1,+\infty] \sqcap_{\text{Itv}} [c,d]) \\
[\![\leqslant]\!]^\sharp_{\text{comp}}\ ([a,b],[c,d]) &= ([a,b] \sqcap_{\text{Itv}} [-\infty, d], [a,+\infty] \sqcap_{\text{Itv}} [c,d]) \\
[\![\neq]\!]^\sharp_{\text{comp}}\ ([a,b],[c,d]) &= ? \textit{ exercise...}
\end{aligned}$$

Abstract interpretation with intervals

```
x := 100;

while 0 < x {

   x := x − 1;
}
```



$$X_1 = [100, 100] \sqcup_{\text{Itv}} \left(X_2 -^{\sharp} [1, 1]\right)$$

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

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

Example : fixpoint iteration

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{aligned}
X_1^0 &= \bot \quad & X_1^{n+1} &= [100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right) \\
X_2^0 &= \bot \quad & X_2^{n+1} &= [1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1} \\
X_3^0 &= \bot \quad & X_3^{n+1} &= [-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}
\end{aligned}
$$

X_1	$\bot$	$\cdots$
X_2	$\bot$	$\cdots$
X_3	$\bot$	$\cdots$

Example : fixpoint iteration

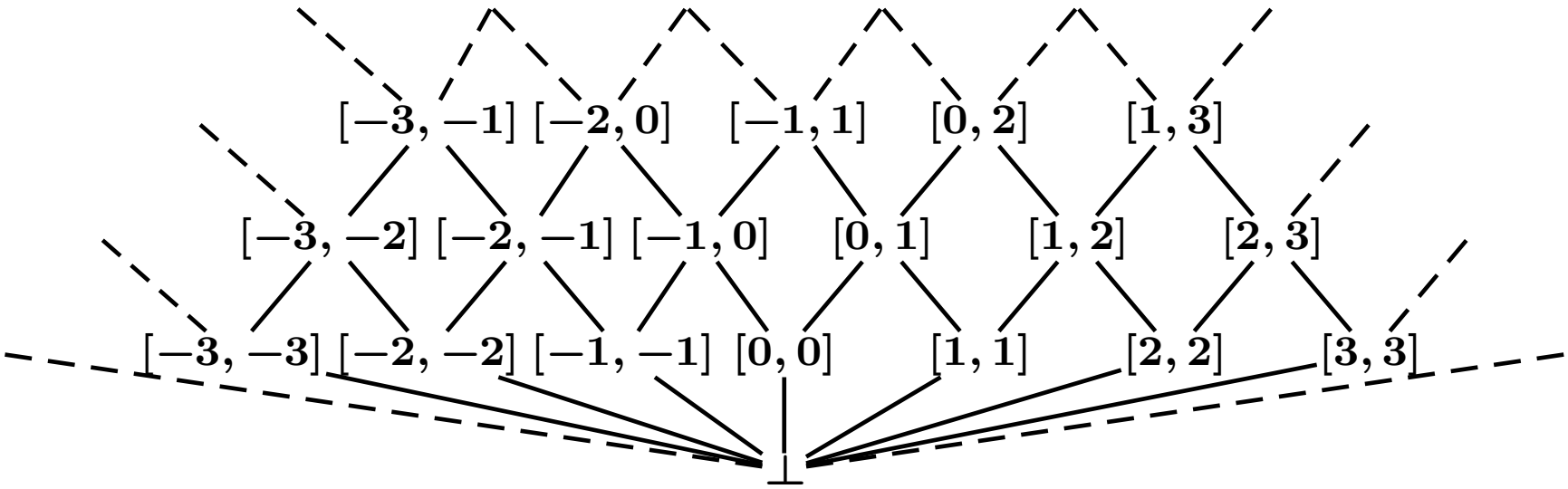
$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{aligned}
X_1^0 &= \bot \quad & X_1^{n+1} &= [100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right) \\
X_2^0 &= \bot \quad & X_2^{n+1} &= [1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1} \\
X_3^0 &= \bot \quad & X_3^{n+1} &= [-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}
\end{aligned}
$$

X_1	$\bot$	$[100,100]$	$[99,100]$	$[98,100]$	$[97,100]$	$\cdots$	$[1,100]$	$[0,100]$
X_2	$\bot$	$[100,100]$	$[99,100]$	$[98,100]$	$[97,100]$	$\cdots$	$[1,100]$	$[1,100]$
X_3	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$	$\bot$	$[0,0]$

Convergence problem



The lattice of intervals has infinite ascending chains.

$$\bot \sqsubset [0,0] \sqsubset [0,1] \sqsubset \cdots \sqsubset [0,n] \sqsubset \cdots .$$

.

Solution : dynamic approximation

- we extrapolate the limit thanks to a widening operator ∇

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

Exercise

Perform an interval analysis of the program

$$
\begin{array}{l}
[^0[x := -1]; \\
\texttt{if}\ {}^1[x < 0]\ \{ \\
\qquad {}^2[y := -3]; \\
\}\ \texttt{else}\ \{ \\
\qquad {}^3[y := 3]; \\
\};]^4
\end{array}
$$

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

Outline

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

$$\mathrm{lfp}(f) \sqsubseteq a$$

.

We may want to over-approximate $\mathrm{lfp}(f)$ when :

- The lattice does not satisfies the ascending chain condition, the iteration $\bot, f(\bot), \ldots, f^n(\bot), \ldots$ 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

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

We will replace it by something of the form

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

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.

Widening : definition

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

- $\forall x, x' \in L, x \sqcup x' \sqsubseteq x \nabla x'$ (implies (i) & (ii))
- If $x^0 \sqsubseteq x^1 \sqsubseteq \ldots$ 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 $x^0 = \bot, x^{n+1} = F(x^n)$

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

Widening : theorem

Theorem

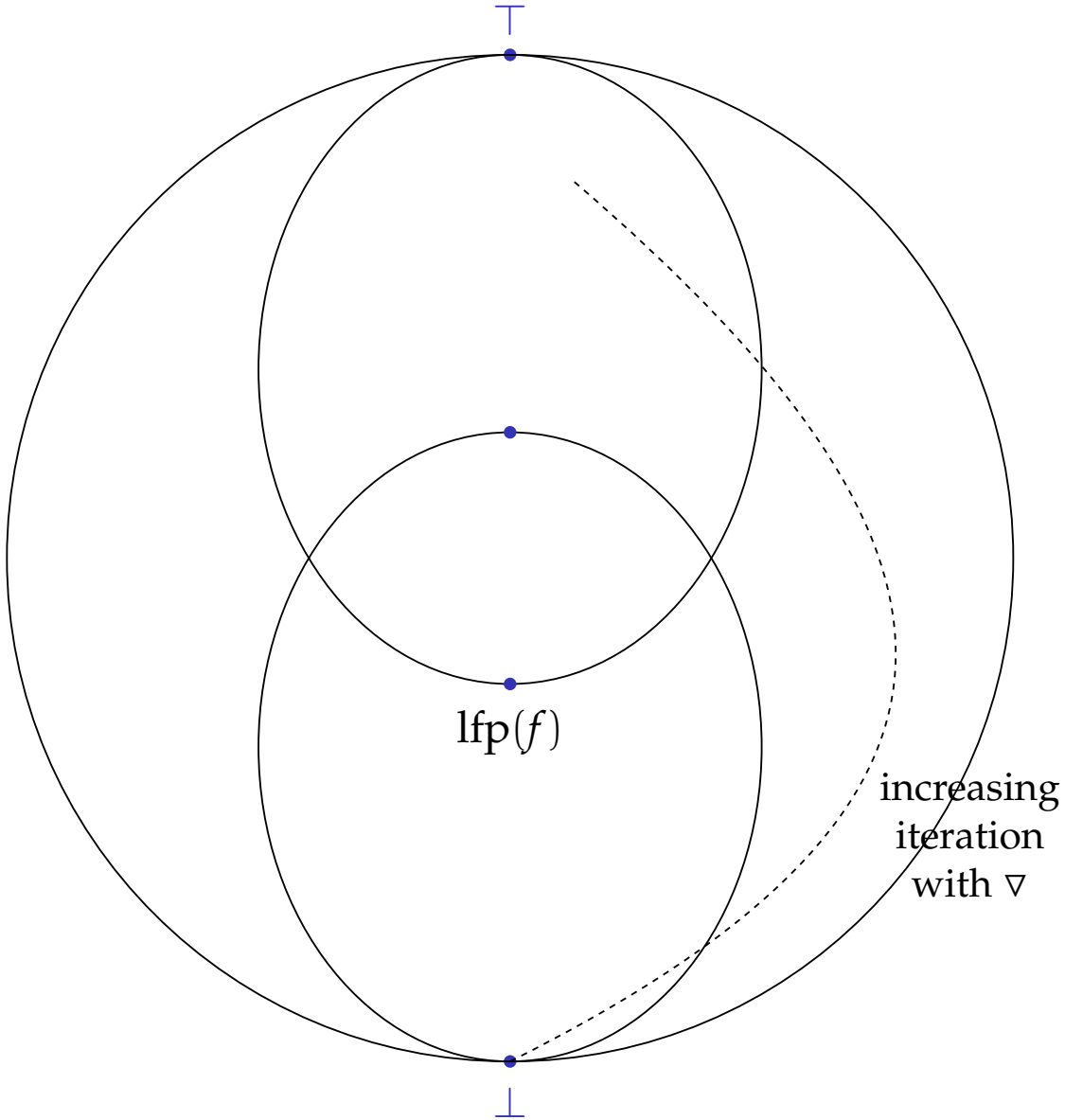
Let

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

Then, the chain $y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$ *stabilizes after a finite number of steps at a post-fixpoint* y *of* F.

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

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 :

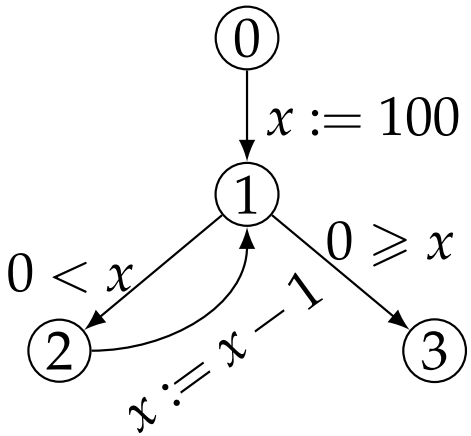
$$\begin{array}{lcl} [a,b]\nabla_{\text{Itv}}[a',b'] & = & [\quad \text{if } a' < a \text{ then } -\infty \text{ else } a, \\ & & \quad\ \text{if } b' > b \text{ then } +\infty \text{ else } b\,] \\ \bot\nabla_{\text{Itv}}[a',b'] & = & [a',b'] \\ I\ \nabla_{\text{Itv}}\ \bot & = & I \end{array}$$

Examples :

$[-3,4]\nabla_{\text{Itv}}[-3,2] = [-3,4]$
$[-3,4]\nabla_{\text{Itv}}[-3,5] = [-3,+\infty]$

Example

```
x := 100;

while 0 < x {

   x := x − 1;
}
```



$$X_1 = [100, 100] \sqcup_{\text{Itv}} \left(X_2 -^{\sharp} [1, 1]\right)$$

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

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

Example : without widening

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{llll}
X_1^0 = \bot & X_1^{n+1} = [100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right) \\
X_2^0 = \bot & X_2^{n+1} = [1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1} \\
X_3^0 = \bot & X_3^{n+1} = [-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}
\end{array}
$$

X_1	$\bot$	$\cdots$
X_2	$\bot$	$\cdots$
X_3	$\bot$	$\cdots$

Example : without widening

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{aligned}
X_1^0 &= \bot \quad & X_1^{n+1} &= [100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right) \\
X_2^0 &= \bot \quad & X_2^{n+1} &= [1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1} \\
X_3^0 &= \bot \quad & X_3^{n+1} &= [-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}
\end{aligned}
$$

X_1	$\bot$	$[100,100]$	$[99,100]$	$[98,100]$	$[97,100]$	$\cdots$	$[1,100]$	$[0,100]$
X_2	$\bot$	$[100,100]$	$[99,100]$	$[98,100]$	$[97,100]$	$\cdots$	$[1,100]$	$[1,100]$
X_3	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$	$\bot$	$[0,0]$

Example : with widening at each node of the cfg

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

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = \bot & X_1^{n+1} = X_1^n \nabla_{\text{Itv}} \left([100,100] \sqcup_{\text{Itv}} (X_2^n -^\sharp [1,1])\right) \\
X_2^0 = \bot & X_2^{n+1} = X_2^n \nabla_{\text{Itv}} \left([1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1}\right) \\
X_3^0 = \bot & X_3^{n+1} = X_3^n \nabla_{\text{Itv}} \left([-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}\right)
\end{array}
$$

X_1	$\bot$
X_2	$\bot$
X_3	$\bot$

Example : with widening at each node of the cfg

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

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = \bot & X_1^{n+1} = X_1^n \nabla_{\text{Itv}} \left([100,100] \sqcup_{\text{Itv}} (X_2^n -^\sharp [1,1])\right) \\
X_2^0 = \bot & X_2^{n+1} = X_2^n \nabla_{\text{Itv}} \left([1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1}\right) \\
X_3^0 = \bot & X_3^{n+1} = X_3^n \nabla_{\text{Itv}} \left([-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}\right)
\end{array}
$$

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

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

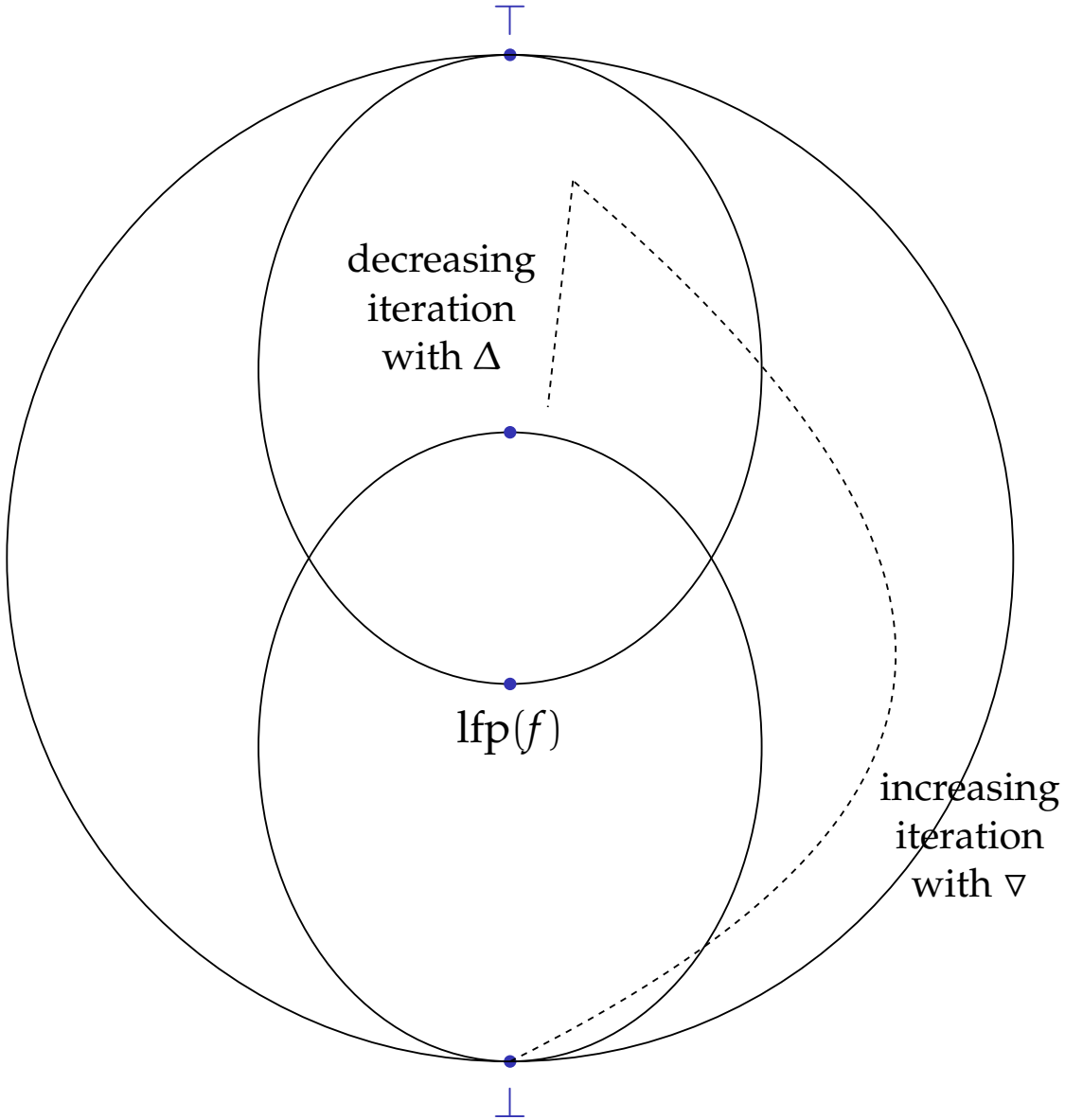
$$\begin{cases} x_0 & = & a \\ x_{k+1} & = & f(x_k) \end{cases}$$

admits for limit $(\bigsqcap\{x_n\})$ the greatest fixpoint of f smaller than a (written $\mathrm{gfp}_a(f)$).

- $\mathrm{lfp}(f) \sqsubseteq \bigsqcap\{x_n\}$.
- *Each intermediate step is a correct approximation :*

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

Scheme


⊤
decreasing
iteration
with Δ
lfp(f)
increasing
iteration
with ∇
⊥

Narrowing : definition

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

- $\forall x, x' \in L, x' \sqsubseteq x \Delta x' \sqsubseteq x$
- If $x^0 \sqsupseteq x^1 \sqsupseteq \ldots$ 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 Δ 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

Intuition : improve infinite bounds.

$$
\begin{array}{rcl}
[a,b]\Delta_{\text{Itv}}[c,d] & = & [\text{if } a = -\infty \text{ then } c \text{ else } a \ ; \ \text{if } b = +\infty \text{ then } d \text{ else } b] \\
I \ \Delta_{\text{Itv}} \ \bot & = & \bot \\
\bot \ \Delta_{\text{Itv}} \ I & = & \bot
\end{array}
$$

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

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = [-\infty,100] & X_1^{n+1} = X_1^n \Delta_{\text{Itv}} \left([100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right)\right) \\
X_2^0 = [-\infty,100] & X_2^{n+1} = X_2^n \Delta_{\text{Itv}} \left([1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1}\right) \\
X_3^0 = [-\infty,0] & X_3^{n+1} = X_3^n \Delta_{\text{Itv}} \left([-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}\right)
\end{array}
$$

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

Example : with narrowing at each node of the cfg

$$
\begin{aligned}
X_1 &= [100,100] \sqcup_{\text{Itv}} \left(X_2 -^\sharp [1,1]\right) \\
X_2 &= [1,+\infty] \sqcap_{\text{Itv}} X_1 \\
X_3 &= [-\infty,0] \sqcap_{\text{Itv}} X_1
\end{aligned}
$$

Iteration strategy : $1 \to 2 \to 3 \to 1 \to 2 \to \cdots$

$$
\begin{array}{ll}
X_1^0 = [-\infty,100] & X_1^{n+1} = X_1^n \Delta_{\text{Itv}} \left([100,100] \sqcup_{\text{Itv}} \left(X_2^n -^\sharp [1,1]\right)\right) \\
X_2^0 = [-\infty,100] & X_2^{n+1} = X_2^n \Delta_{\text{Itv}} \left([1,+\infty] \sqcap_{\text{Itv}} X_1^{n+1}\right) \\
X_3^0 = [-\infty,0] & X_3^{n+1} = X_3^n \Delta_{\text{Itv}} \left([-\infty,0] \sqcap_{\text{Itv}} X_1^{n+1}\right)
\end{array}
$$

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 with $f_1, \ldots, f_n$ monotone.

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

Standard iteration :

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

Standard iteration with widening :

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

The particular case of an equation system

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

It is sufficient (and generally more precise) to use ∇ for a selection of index W provided that each cycle in the system has at least one point in W.

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

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

The need for relational program analysis

Consider the program

```
i := 1;
t := 0;
while i < 100 {
   t := t + 1;
   i := i + 2;
}
```

where `t` is inserted to argue termination of the loop.

Interval analysis with widening and narrowing will

- find precise bounds for `i` ($[1, 100]$
- but not for `t` (only finds $[0, \infty]$).

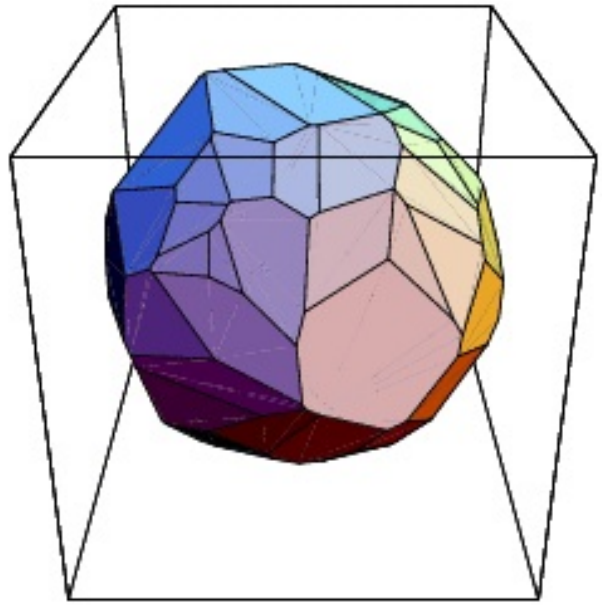
Need to know that the two variables are related : $\texttt{i} = 2\texttt{t} + 1$.

Polyhedral abstract interpretation

Polyhedral analysis seeks to discover invariants of **linear equality and inequality** relations (such as $x = y$ or $x \leqslant 2y + z$) among the variables of an imperative program.

A convex polyhedron can be defined

- algebraically as the set of solutions of a system of linear inequalities.
- geometrically, as a finite intersection of half-spaces.

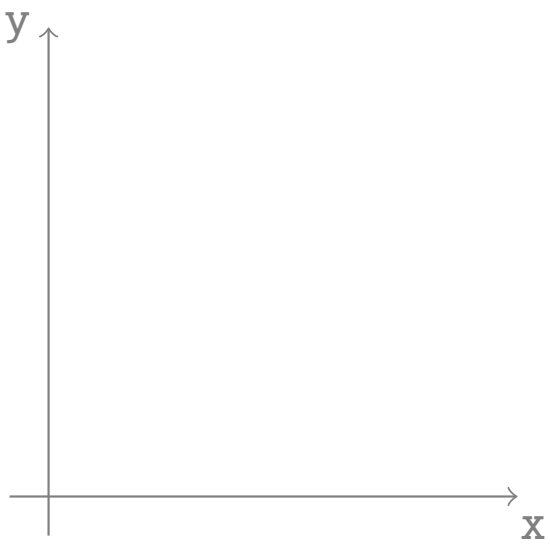


The classical reference :

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

Polyhedral analysis

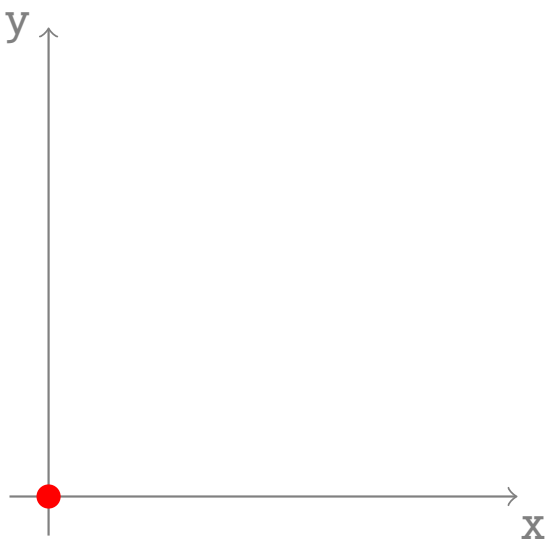
State properties are over-approximated by convex polyhedra in $\mathbb{Q}^2$.




```
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 $\mathbb{Q}^2$.



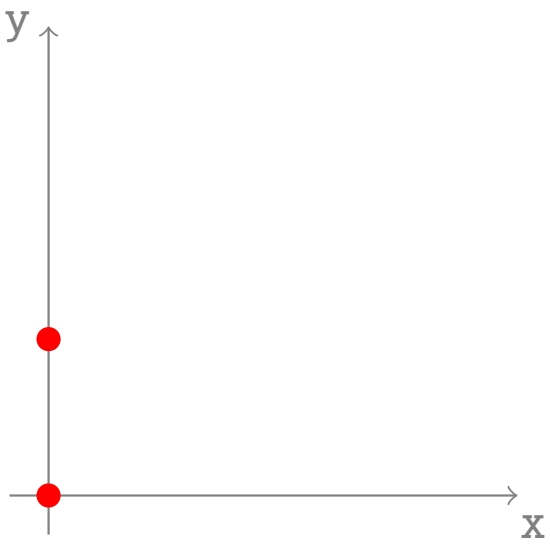

```
x = 0; y = 0;
```
$\{x = 0 \wedge y = 0\}$

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

```
while (x<6) {
  if (?) {
```
$\{x = 0 \wedge y = 0\}$

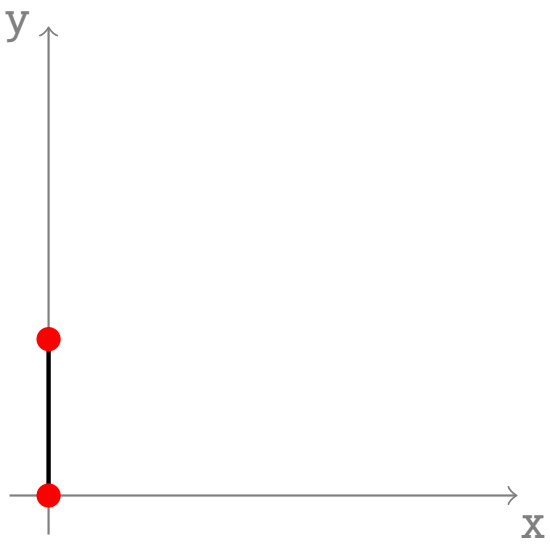
```
    y = y+2;
```
$\{x = 0 \wedge y = 2\}$

```
  };
```
$\{x = 0 \wedge y = 0\} \uplus \{x = 0 \wedge y = 2\}$

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

```
while (x<6) {
  if (?) {
```
$\{x = 0 \wedge y = 0\}$

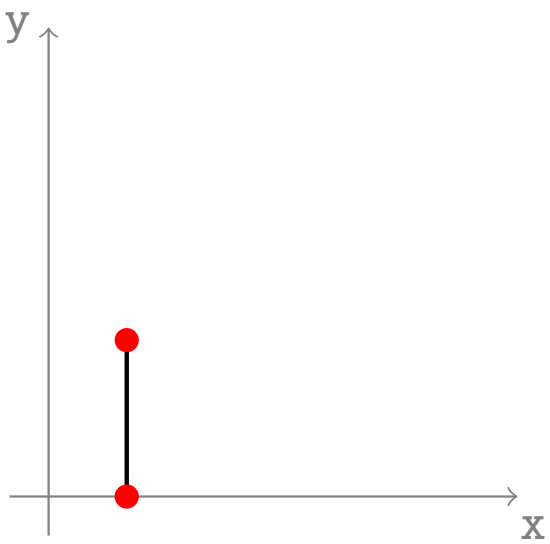
```
    y = y+2;
```
$\{x = 0 \wedge y = 2\}$

```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  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;
```
$\{x = 0 \wedge y = 2\}$

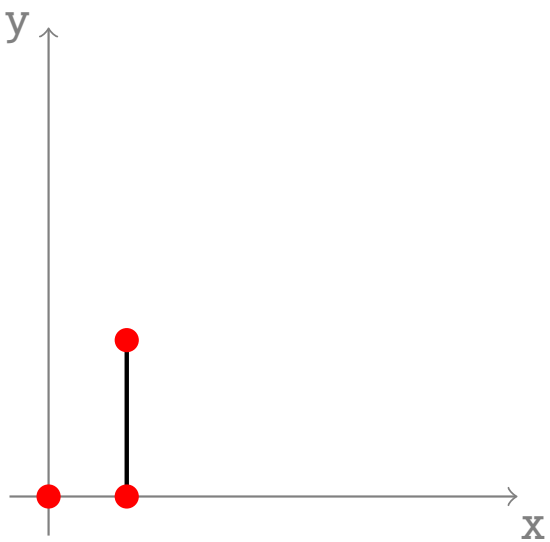
```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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



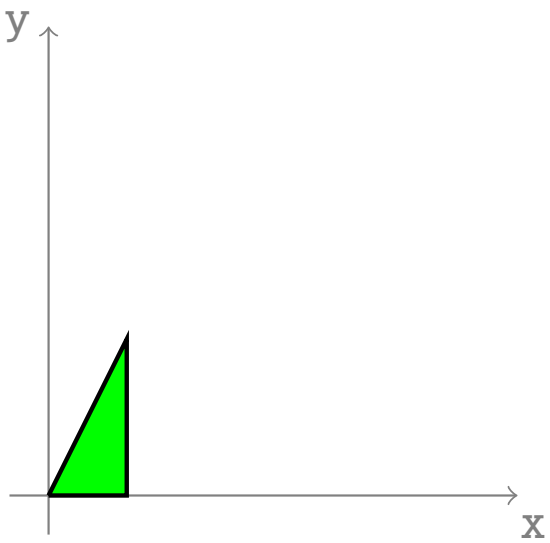

```
x = 0; y = 0;
```
$\{x = 0 \wedge y = 0\} \uplus \{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

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

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$
```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x = 0 \wedge y = 0\}$

```
    y = y+2;
```
$\{x = 0 \wedge y = 2\}$

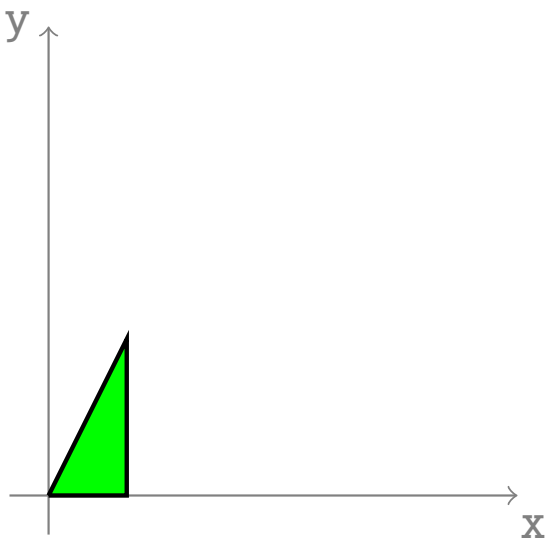
```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x = 0 \wedge y = 2\}$

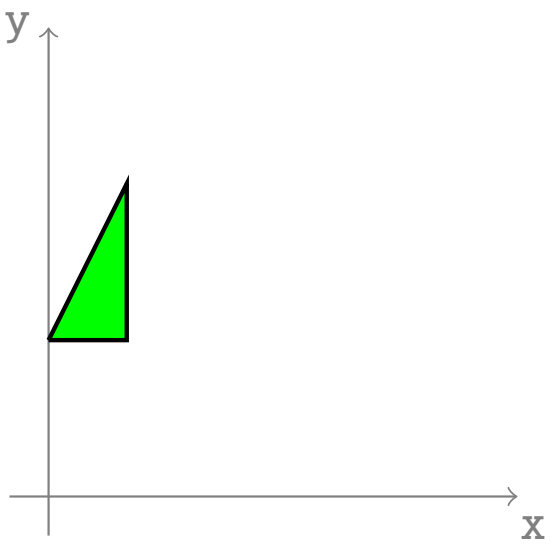
```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

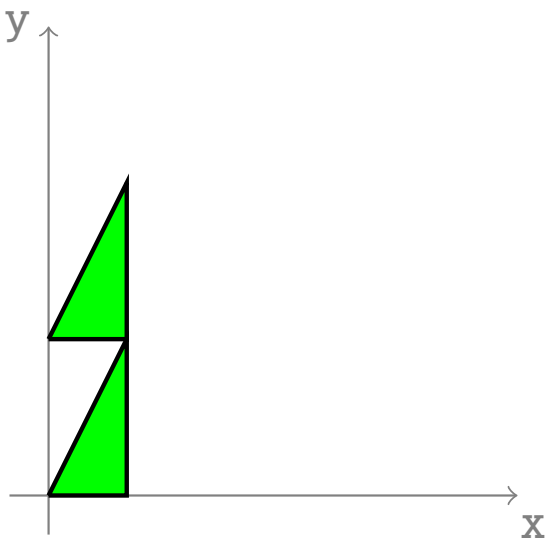
```
  };
```
$\{x = 0 \wedge 0 \leqslant y \leqslant 2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

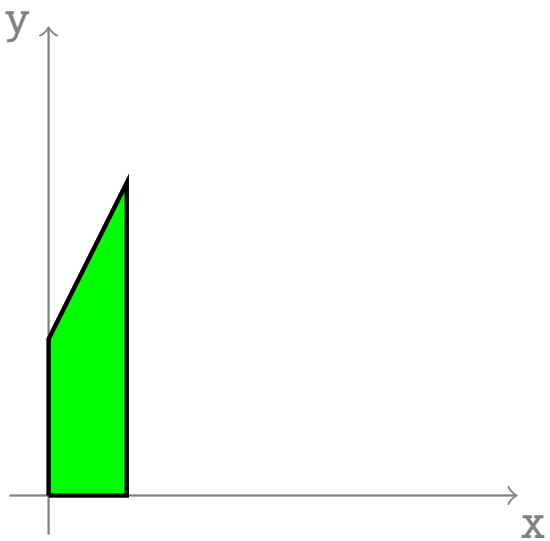
```
  };
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\} \uplus \{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

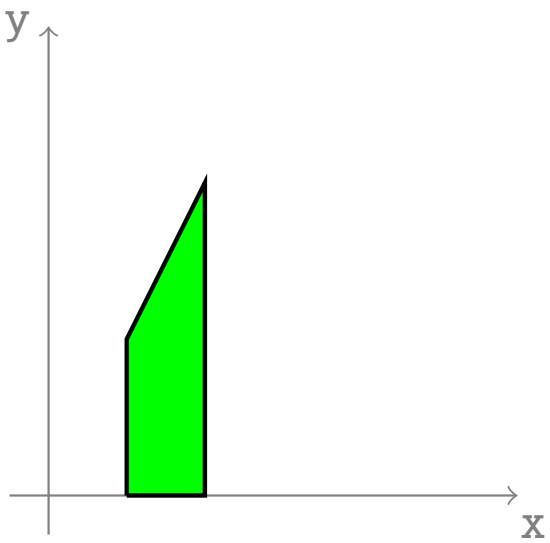
```
  };
```
$\{0 \leqslant x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x+2\}$

```
  x = x+1;
```
$\{x = 1 \wedge 0 \leqslant y \leqslant 2\}$

```
}
```

Polyhedral analysis

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




```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x + 2\}$

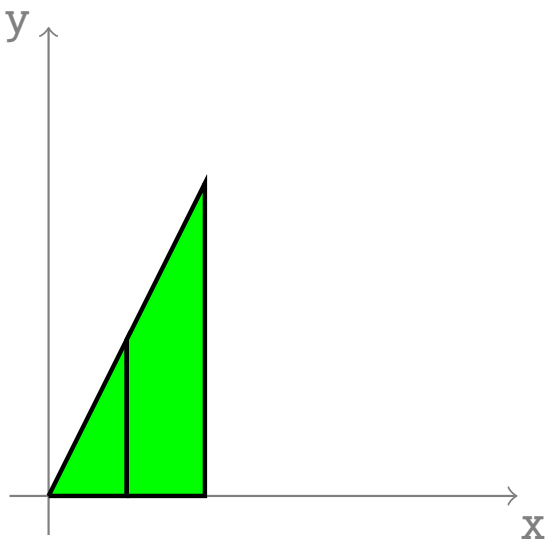
```
  };
```
$\{0 \leqslant x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x + 2\}$

```
  x = x+1;
```
$\{1 \leqslant x \leqslant 2 \wedge 0 \leqslant y \leqslant 2x\}$

```
}
```

Polyhedral analysis

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

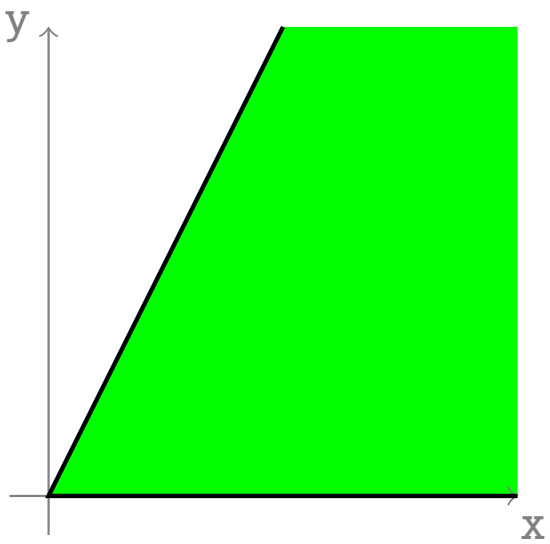


At loop headers, we use heuristics (widening) to ensure finite convergence.

```
x = 0; y = 0;
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$
$\nabla \{x \leqslant 2 \wedge 0 \leqslant y \leqslant 2x\}$
```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$
```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$
```
  };
```
$\{0 \leqslant x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x+2\}$
```
  x = x+1;
```
$\{1 \leqslant x \leqslant 2 \wedge 0 \leqslant y \leqslant 2x\}$
```
}
```

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;
```
$\{0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x\}$

```
    y = y+2;
```
$\{x \leqslant 1 \wedge 2 \leqslant y \leqslant 2x+2\}$

```
  };
```
$\{0 \leqslant x \leqslant 1 \wedge 0 \leqslant y \leqslant 2x+2\}$

```
  x = x+1;
```
$\{1 \leqslant x \leqslant 2 \wedge 0 \leqslant y \leqslant 2x\}$

```
}
```

Polyhedral analysis

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

By propagation we obtain a post-fixpoint

```
x = 0; y = 0;
```
$\{0 \leqslant y \leqslant 2x\}$

```
while (x<6) {
  if (?) {
```
$\{0 \leqslant y \leqslant 2x \wedge x \leqslant 5\}$

```
    y = y+2;
```
$\{2 \leqslant y \leqslant 2x+2 \wedge x \leqslant 5\}$

```
  };
```
$\{0 \leqslant y \leqslant 2x+2 \wedge 0 \leqslant x \leqslant 5\}$

```
  x = x+1;
```
$\{0 \leqslant y \leqslant 2x \wedge 1 \leqslant x \leqslant 6\}$

```
}
```
$\{0 \leqslant y \leqslant 2x \wedge 6 \leqslant x\}$

Polyhedral analysis

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

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

```
x = 0; y = 0;
```
$\{0 \leqslant y \leqslant 2x \ \wedge x \leqslant 6\}$

```
while (x<6) {
  if (?) {
```
$\{0 \leqslant y \leqslant 2x \ \wedge \ x \leqslant 5\}$

```
    y = y+2;
```
$\{2 \leqslant y \leqslant 2x + 2 \ \wedge \ x \leqslant 5\}$

```
  };
```
$\{0 \leqslant y \leqslant 2x + 2 \ \wedge \ 0 \leqslant x \leqslant 5\}$

```
  x = x+1;
```
$\{0 \leqslant y \leqslant 2x \ \wedge \ 1 \leqslant x \leqslant 6\}$

```
}
```
$\{0 \leqslant y \leqslant 2x \ \wedge \ 6 = x\}$

Polyhedral analysis

A more complex example.

The analysis accepts to replace some constants by parameters.

```
x = 0; y = A;
```
$\{A \leqslant y \leqslant 2x + A \wedge x \leqslant N\}$

```
while (x<N) {
  if (?) {
```
$\{A \leqslant y \leqslant 2x + A \ \wedge \ x \leqslant N - 1\}$

```
    y = y+2;
```
$\{A + 2 \leqslant y \leqslant 2x + A + 2 \ \wedge \ x \leqslant N - 1\}$

```
  };
```
$\{A \leqslant y \leqslant 2x + A + 2 \ \wedge \ 0 \leqslant x \leqslant N - 1\}$

```
  x = x+1;
```
$\{A \leqslant y \leqslant 2x + A \ \wedge \ 1 \leqslant x \leqslant N\}$

```
}
```
$\{A \leqslant y \leqslant 2x + A \ \wedge \ N = x\}$

The four polyhedra operations

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

```
x = 0; y = 0;
```
$P_0 = [\![\mathtt{y} := \mathtt{0}]\!]\,[\![\mathtt{x} := \mathtt{0}]\!]\,(\mathbb{Q}^2) \,\nabla\, P_4$
```
while (x<6) {
  if (?) {
```
$P_1 = P_0 \cap \{x < 6\}$
```
    y = y+2;
```
$P_2 = [\![\mathtt{y} := \mathtt{y} + \mathtt{2}]\!]\,(P_1)$
```
  };
```
$P_3 = P_1 \uplus P_2$
```
  x = x+1;
```
$P_4 = [\![\mathtt{x} := \mathtt{x} + \mathtt{1}]\!]\,(P_3)$
```
}
```
$P_5 = P_0 \cap \{x \geqslant 6\}$

More about abstraction of assignments

Distinguish between two types of assignments `x := exp`

- invertible assignments, where `x` appears in `exp`
  Example : `x = 2x + y`
- non-invertible assignments, when the new value of `x` does not depend on its old value.
  Example : `x = y + 42`

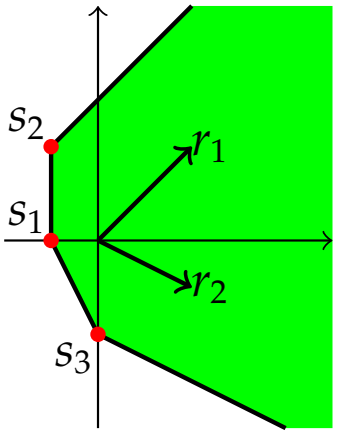
Invertible assignments can be abstracted precisely in two steps :

1. express the old value of `x` as an expression involving the new value (written `x'`)
   Example : for `x := x + 1`, we have `x'-1 = x`
2. in the constraints, replace all occurrences of `x` by the inverted expression for `x`.
   Example : $y \geqslant 2x$ becomes $y \geqslant 2(x-1)$.

For non-invertible assignments : all information about old `x` is lost, so we remove all constraints involving `x`, and add $x = exp$.

Library for manipulating polyhedra

- Parma Polyhedra Library [1] (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

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

  - by set of generators

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

  - 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/`

Other relational domains

Polyhedral analysis uses all linear relations ($\bigwedge_j \sum_i a_{ij} x_i \geqslant c_j$).
It is precise but has exponential complexity.

Other examples of (weakly) relational analyses

- linear equalities ($x = \sum_i a_i x_i$)
- zones and octagons.

Octagons : restrict constraints to be of form $\pm X_1 \pm X_2 \leqslant C$.

- At most two variables are related in one constraint.
- Only allowed coefficients are -1,1.

Efficient polynomial-time operations and good precision.

Example (Miné p. 131) : least upper bound of two boxes with intervals and with octagons :