Non-Relational Numerical Abstract Domains

MPRI 2–6: Abstract Interpretation,
application to verification and static analysis

Antoine Miné

year 2015–2016

course 04
30 September 2015

Outline

- Some applications of numerical domains
- Generalities, notations
- Presentation of a few numerical abstract domains (non-relational)
  - sign domains
  - constant domain
  - interval domain
  - simple congruence domain
- Reduced products of domains
- Bibliography

Selected applications of numerical domains

Invariant discovery

Goal: find intermittent numerical invariants

(at each program point, properties of numerical variables true for all executions)

Example

```
X:=[0,10]; Y:=100;

while X>=0 do
    // loop invariant?
  X:=X-1;

  Y:=Y+10

done
// value of X and Y?
```

Invariant discovery

Goal: find intermittent numerical invariants

(at each program point, properties of numerical variables true for all executions)

Example

```
X:=[0,10]; Y:=100;
   // X ∈ [0,10], Y = 100
while X>=0 do
      // X ∈ [0,10], Y ∈ [100,200]
   X:=X-1;
      // X ∈ [−1,9], Y ∈ [100,200]
   Y:=Y+10
      // X ∈ [−1,9], Y ∈ [110,210]
done
// X = −1, Y ∈ [110,210]
```

Variable bounds

Invariant discovery

Hope: find **the strongest** intermittent numerical invariants

(at each program point, **the strongest** properties of numerical variables true for all executions)

Example

$X := [0,10];\ Y := 100;$

$\quad // \ X \in [0,10],\ Y = 100$

while $X >= 0$ do

$\qquad // \ X \in [0,10],\ 10X + Y \in [100,200] \cap 10\mathbb{Z}$

$\quad X := X - 1;$

$\qquad // \ X \in [-1,9],\ 10X + Y \in [90,190] \cap 10\mathbb{Z}$

$\quad Y := Y + 10$

$\qquad // \ X \in [-1,9],\ 10X + Y \in [100,200] \cap 10\mathbb{Z}$

done

$// \ X = -1,\ Y \in [110,210] \cap 10\mathbb{Z}$

Variable bounds, linear relations and congruences

Application: proof of absence of run-time error

delay line, in C

```
int delay[10], i;
for (i=10; i>0; i=i-1)
    delay[i-1] = 0;
while (1) {
    int y = delay[i];
    delay[i] = input();
    i = i+1;
    if (i>=10) i = 0;
    /* use y */
}
```

Some operations are undefined or dangerous:

- arithmetic operations can overflow
- arrays can be accessed out of bounds

Application: proof of absence of run-time error

delay line, in C

```
int delay[10], i;
for (i=10; i>0; ⟨i − 1 ∈ [−2^31, 2^31 − 1]⟩ i=i-1)
    ⟨i − 1 ∈ [0, 9]⟩ delay[i-1] = 0;
while (1) {
    int y = ⟨i ∈ [0, 9]⟩ delay[i];
    ⟨i ∈ [0, 9]⟩ delay[i] = input();
    ⟨i + 1 ∈ [−2^31, 2^31 − 1]⟩ i = i+1;
    if (i>=10) i = 0;
    /* use y */
}
```

To prove the absence of run-time error:

- insert verification conditions $\langle\cdot\rangle$ ensuring error-freedom

Application: proof of absence of run-time error

delay line, in C

```
int delay[10], i;
for (i=10; i>0; (i ∈ [1,10]) ⟨i − 1 ∈ [−2^31, 2^31 − 1]⟩ i=i-1)
    (i ∈ [1,10]) ⟨i − 1 ∈ [0,9]⟩ delay[i-1] = 0;
(i = 0) while (1) {
    int y = (i ∈ [0,9]) ⟨i ∈ [0,9]⟩ delay[i];
    (i ∈ [0,9]) ⟨i ∈ [0,9]⟩ delay[i] = input();
    (i ∈ [0,9]) ⟨i + 1 ∈ [−2^31, 2^31 − 1]⟩ i = i+1;
    (i ∈ [1,10]) if (i>=10) i = 0 (i ∈ [0,9]);
    /* use y */
}
```

To prove the absence of run-time error:

- insert verification conditions $\langle\cdot\rangle$ ensuring error-freedom
- infer invariants $(\cdot)$
- check in the abstract that the invariants imply the conditions

  (e.g., reduces to interval inclusion in the interval domain)

Forward–backward analysis

sign function

```
X:=[-100,100];
if X=0 then Z:=0 else
  Y:=X;
  if Y < 0 then Y:=-Y;
  Z:=X/Y
fi
```

Forward–backward analysis

sign function

```
X:=[-100,100]; (X ∈ [−100, 100])
if X=0 then Z:=0 else (X ∈ [−100, 100])
  Y:=X; (X, Y ∈ [−100, 100])
  if Y < 0 then Y:=-Y; (X ∈ [−100, 100], Y ∈ [0, 100])
  Z:=X/Y (X ∈ [−100, 100], Y ∈ [0, 100])
fi
```

Forward interval analysis
(possible division by 0)

Forward–backward analysis

sign function

```
X:=[-100,100]; (⊥)
if X=0 then Z:=0 else (X = 0)
  Y:=X; (Y = 0)
  if Y < 0 then Y:=-Y; (Y = 0)
  Z:=X/Y (Y = 0)
fi
```

Backward interval analysis

- infer (tight) necessary conditions on inputs to reach a given point in a given state ($Y = 0$ at the end of the program)
- refine and focus the result of a forward analysis (prove the absence of division by zero) [Bour93b]

Relation analysis

store the maximum of X,Y,0 into Z

```
max(X,Y,Z)

  Z :=X ;
  if Y  > Z  then Z :=Y ;
  if Z  < 0 then Z :=0;
```

Relation analysis

store the maximum of X,Y,0 into Z'

```
max(X,Y,Z)
  X':=X; Y':=Y; Z':=Z;
  Z':=X';
  if Y' > Z' then Z':=Y';
  if Z' < 0 then Z':=0;
```

- add and rename variables: keep a copy of input values

Relation analysis

store the maximum of X,Y,0 into Z'

```
max(X,Y,Z)
  X':=X; Y':=Y; Z':=Z;
  Z':=X';
  if Y' > Z' then Z':=Y';
  if Z' < 0 then Z':=0;
```
$(Z' \geq X \wedge Z' \geq Y \wedge Z' \geq 0 \wedge X' = X \wedge Y' = Y)$

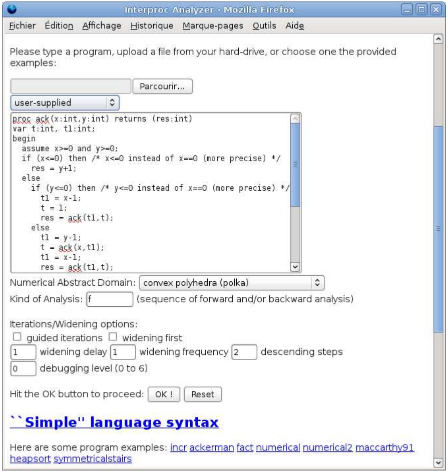
- add and rename variables: keep a copy of input values
- infer a relation between input values (X,Y,Z) and current values (X′,Y′,Z′)

Applications: procedure summaries, modular analyses. [Anco10]

Academic implementation: Apron and Interproc

Apron: library of numerical abstractions [Jean09]

Interproc: on-line analyzer for a toy language, based on Apron



`http://pop-art.inrialpes.fr/interproc/interprocweb.cgi`

Applications to non-numerical analyses

Pointer offset analysis

pointer arithmetic

```
float* p = q;
for (i=0; i<10; i++)
  if (...)  p++;
```

$\rightsquigarrow$

offset arithmetic

```
unsigned off_p = off_q;
for (i=0; i<10; i++)
  if (...)  off_p += 4;
```

$(off_q \leq off_p \leq off_q + 4 \times i + 4)$

In C, pointers can be viewed as symbolic integers with:

- a symbolic base
- an integer offset (off_p, off_q)

[Mine06]

String analysis for C

pointers and buffers

```
char buf[20];
char* p;

strcpy(buf, "Hello");
p = buf+5;

strcpy(p, " world!");
```

In C, strings are pointers to arrays of `char`, terminated by 0:

- no explicit information on available space (buffer length)
- no explicit length information (position of 0)
- aliasing is possible

$\Longrightarrow$ source of many programming errors

String analysis for C

pointers and buffers

char buf[20]; $(alloc_{buf} = 20)$
char* p;
$\langle alloc_{buf} \geq 6 \rangle$
strcpy(buf, "Hello"); $(len_{buf} = 5)$
p = buf+5; $(stride_{p-buf} = 5, len_p = len_{buf} - 5, alloc_p = alloc_{buf} - 5)$
$\langle alloc_p \geq 8 \rangle$
strcpy(p, " world!"); $(len_p = 7, len_{buf} = len_p + stride_{p-buf})$

Analysis of correctness: [Dor01]

- instrument the program with integer variables ($alloc_p$, len_p, $stride_{p-q}$)
- add code to update the variables $(\cdot)$
- add safety assertions $\langle \cdot \rangle$
- infer invariants and prove that the assertions hold

Memory shape analysis

list creation and copy into an array

```
cell *x, *head = NULL;
for (i=0; i<n; i++) {
  x = alloc();
  x->next = head; head = x;
}
```
$(k \in [0, n-1] \wedge head(\text{->}next)^k\text{->}data = 0)$
```
for (i=0, x=head; x; x=x->next, i++)
  a[i] = x->data;
```
$(k \in [0, n-1] \wedge a[k] = head(\text{->}next)^k\text{->}data)$

Numerical analysis on:

- program variables: i, n, and
- instrumentation variables: k, $head(\text{->}next)^k\text{->}data$, `a[`$k$`]`

[Vene02]

Cost analysis

selection sort

```
cost = 0;
for i=0 to n-2 do
  for j=i+1 to n-1 do
    if tab[i] > tab[j] then swap(tab[i],tab[j]);
    cost = cost+1
  done
done
```

To count the maximum number of instructions:

- instrument the program with a counter

Cost analysis

selection sort

```
cost = 0;
for i=0 to n-2 do (cost = i × n − i × (i + 1)/2)
  for j=i+1 to n-1 do (cost = i × n − i × (i + 1)/2 + j − i − 1)
    if tab[i] > tab[j] then swap(tab[i],tab[j]);
    cost = cost+1
  done
done
(cost = (n + 1) × (n − 2)/2)
```

To count the maximum number of instructions:

- instrument the program with a counter
- infer loop and exit invariants $(\cdot)$

Dependency analysis for array indices

multiplication of polynomials

```
for i=1 to n do
  for j=1 to n do
    v := r[i+j] •;
    ♠ r[i+j] := v + a[i] * b[j];
    t := t+1
  done
done
```

Can a read at • depend on a previous write from ♠?

- add a global counter t (allows expressing temporal properties)
- infer an invariant set $X \in \mathbb{Z}^3$ for t, i, j
- check $\exists((t,i,j),(t',i',j')) \in X \times X,\ t > t',\ i+j = i'+j'$

Information used by compilers to enable loop transformations [Girb06].

Generalities and notations

Syntax

Expression syntax

Toy language:

- fixed, finite set of variables $\mathbb{V}$,
- one datatype: scalars in $\mathbb{I}$, with $\mathbb{I} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$
  (and later, floating-point numbers $\mathbb{F}$)
- no procedure

arithmetic expressions:

$$
\begin{array}{lcll}
\texttt{exp} & ::= & \texttt{V} & \text{variable } \texttt{V} \in \mathbb{V} \\
& | & -\texttt{exp} & \text{negation} \\
& | & \texttt{exp} \diamond \texttt{exp} & \text{binary operation: } \diamond \in \{+, -, \times, /\} \\
& | & [c, c'] & \text{constant range, } c, c' \in \mathbb{I} \cup \{\pm\infty\} \\
& & & c \text{ is a shorthand for } [c, c]
\end{array}
$$

Programs (structured syntax)

programs: as syntax trees

prog ::=		
$\mid$	`V := exp`	assignment
$\mid$	`if exp` $\bowtie 0$ `then prog else prog fi`	test
$\mid$	`while exp` $\bowtie 0$ `do prog done`	loop
$\mid$	`prog; prog`	sequence
$\mid$	ε	no-op

comparison operators: $\bowtie \in \{=,<,>,<=,>=,<>\}$.

Programs (as control-flow graphs)

commands:

$$
\begin{array}{lcll}
\texttt{com} & ::= & \texttt{V := exp} & \text{assignment into } \texttt{V} \in \mathbb{V} \\
 & | & \texttt{exp} \bowtie 0 & \text{test, } \bowtie \in \{=,<,>,<=,>=,<>\}
\end{array}
$$

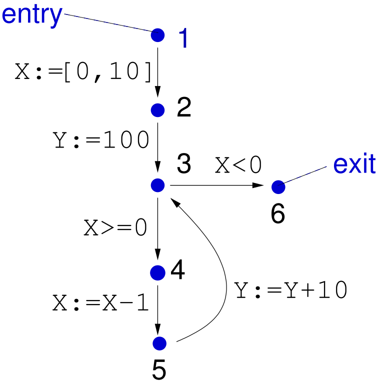
programs: as control-flow graphs

$$
P \stackrel{\text{def}}{=} (L, e, x, A)
\quad\left|\quad
\begin{array}{ll}
L & \text{program points (labels)} \\
e & \text{entry point: } e \in L \\
x & \text{exit point: } x \in L \\
A & \text{arcs: } A \subseteq L \times \texttt{com} \times L
\end{array}\right.
$$

Example

```
¹X:=[0,10];²
 Y:=100;
 while ³X>=0 do⁴
   X:=X-1;⁵
   Y:=Y+10
 done⁶
```

structured program



control flow graph

Concrete semantics

Forward concrete semantics

Semantics of expressions: $\mathsf{E}[\![\, e \,]\!] : (\mathbb{V} \to \mathbb{I}) \to \mathcal{P}(\mathbb{I})$

The evaluation of e in ρ gives a set of values:

$$
\begin{array}{lcl}
\mathsf{E}[\![\, [c,c'] \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, x \in \mathbb{I} \mid c \leq x \leq c' \,\} \\
\mathsf{E}[\![\, \mathtt{V} \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, \rho(\mathtt{V}) \,\} \\
\mathsf{E}[\![\, -e \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, -v \mid v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\} \\
\mathsf{E}[\![\, e_1 + e_2 \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, v_1 + v_2 \mid v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \,\} \\
\mathsf{E}[\![\, e_1 - e_2 \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, v_1 - v_2 \mid v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \,\} \\
\mathsf{E}[\![\, e_1 \times e_2 \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, v_1 \times v_2 \mid v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \,\} \\
\mathsf{E}[\![\, e_1 / e_2 \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, v_1 / v_2 \mid v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho,\ v_2 \neq 0 \,\}
\end{array}
$$

Forward concrete semantics (cont.)

Semantics of commands: $\mathsf{C}[\![\, c \,]\!] : \mathcal{P}(\mathbb{V} \to \mathbb{I}) \to \mathcal{P}(\mathbb{V} \to \mathbb{I})$

A transfer function for c defines a relation on environments:

$$\mathsf{C}[\![\, \mathtt{V} := e \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{\, \rho[\, \mathtt{V} \mapsto v \,] \mid \rho \in \mathcal{X},\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\}$$
$$\mathsf{C}[\![\, e \bowtie 0 \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{\, \rho \mid \rho \in \mathcal{X},\ \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho,\ v \bowtie 0 \,\}$$

It relates the environments after the execution of a command to the environments before.

Complete join morphism: $\mathsf{C}[\![\, c \,]\!]\, \mathcal{X} = \bigcup_{\rho \in \mathcal{X}} \mathsf{C}[\![\, c \,]\!]\, \{\, \rho \,\}$.

Forward concrete semantics (cont.)

Semantics of programs: $\mathsf{P}[\![\,(L,e,x,A)\,]\!] : L \to \mathcal{P}(\mathbb{V} \to \mathbb{I})$

$\mathsf{P}[\![\,(L,e,x,A)\,]\!]\,\ell$ is the most precise invariant at $\ell \in L$.

It is the smallest solution of a recursive equation system $(\mathcal{X}_\ell)_{\ell \in L}$:

Semantic equation system

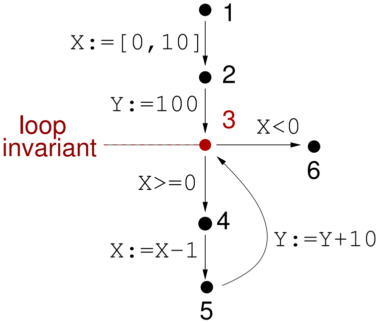
$$\begin{array}{lcll} \mathcal{X}_e & & & \text{(given initial state)} \\ \mathcal{X}_{\ell \neq e} & = & \bigcup_{(\ell',c,\ell) \in A} \mathsf{C}[\![\,c\,]\!]\,\mathcal{X}_{\ell'} & \text{(transfer function)} \end{array}$$

Tarski's Theorem: this smallest solution exists and is unique.

- $\mathcal{D} \stackrel{\text{def}}{=} (\mathcal{P}(\mathbb{V} \to \mathbb{I}), \subseteq, \cup, \cap, \emptyset, (\mathbb{V} \to \mathbb{I}))$ is a complete lattice,
- each $M_\ell : \mathcal{X}_\ell \mapsto \bigcup_{(\ell',c,\ell) \in A} \mathsf{C}[\![\,c\,]\!]\,\mathcal{X}_{\ell'}$ is monotonic in $\mathcal{D}$.

  $\Rightarrow$ the solution is the least fixpoint of $(M_\ell)_{\ell \in L}$.

Forward concrete semantics (example)



control flow graph

$$\left\{\begin{array}{l}
\mathcal{X}_1 = (\{\,\mathtt{X},\mathtt{Y}\,\} \to \mathbb{Z}) \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 \\
\mathcal{X}_3 = \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\
\qquad \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3
\end{array}\right.$$

equation system

Loop invariant:

$$\mathcal{X}_3 = \{\, \rho \mid \rho(\mathtt{X}) \in [0,10],\ 10\rho(\mathtt{X}) + \rho(\mathtt{Y}) \in [100,200] \cap 10\mathbb{Z} \,\}$$

Resolution

Resolution by increasing iterations:

$$\begin{cases} \mathcal{X}_e^0 & \stackrel{\text{def}}{=} & \mathcal{X}_e \\ \mathcal{X}_{\ell\neq e}^0 & \stackrel{\text{def}}{=} & \emptyset \end{cases} \qquad \begin{cases} \mathcal{X}_e^{n+1} & \stackrel{\text{def}}{=} & \mathcal{X}_e \\ \mathcal{X}_{\ell\neq e}^{n+1} & \stackrel{\text{def}}{=} & \displaystyle\bigcup_{(\ell',c,\ell)\in A} \mathsf{C}[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^n \end{cases}$$

Converges in ω iterations to a least solution,
because each $\mathsf{C}[\![\, c \,]\!]$ is continuous in the CPO $\mathcal{D}$.

(Kleene fixpoint theorem)

Resolution (example)

iteration 0

$$\begin{cases} \mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\ \mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & \emptyset \\ \mathcal{X}_3 = \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 & \emptyset \\ \mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \emptyset \\ \mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \emptyset \\ \mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \emptyset \end{cases}$$

Resolution (example)

iteration 1

$$\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} & \emptyset \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \emptyset \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \emptyset \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \emptyset
\end{cases}$$

Resolution (example)

$$\begin{cases} \mathcal{X}_1 = \mathbb{Z}^2 \\ \mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 \\ \mathcal{X}_3 = \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \qquad \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \\ \mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 \\ \mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 \\ \mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 \end{cases}$$

iteration 2

$\mathbb{Z}^2$

$[0,10] \times \mathbb{Z}$

$\{\,(0,100), \ldots, (10,100)\,\}$

$\emptyset$

$\emptyset$

$\emptyset$

Resolution (example)

$$\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 \\
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3
\end{cases}$$

iteration 3

$\mathbb{Z}^2$

$[0,10] \times \mathbb{Z}$

$\{\, (0,100), \ldots, (10,100) \,\}$

$\{\, (0,100), \ldots, (10,100) \,\}$

$\emptyset$

$\emptyset$

Resolution (example)

$$\begin{cases}
& \text{iteration 4} \\
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} & \{\, (0,100), \ldots, (10,100) \,\} \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \{\, (0,100), \ldots, (10,100) \,\} \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \{\, (-1,100), \ldots, (9,100) \,\} \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \emptyset
\end{cases}$$

Resolution (example)

$$
\begin{cases}
 & \text{iteration 5} \\
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\
\mathcal{X}_3 = \;\; \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup & \{\, (0,100), \ldots, (10,100), \\
\qquad\quad \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 & \;\; (-1,110), \ldots, (9,110) \,\} \\
\mathcal{X}_4 = \;\; \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \{\, (0,100), \ldots, (10,100) \,\} \\
\mathcal{X}_5 = \;\; \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \{\, (-1,100), \ldots, (9,100) \,\} \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \emptyset
\end{cases}
$$

Resolution (example)

$$
\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\[2ex]
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\[2ex]
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \;(-1,110), \ldots, (9,110) \,\} \end{array} \\[2ex]
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \;(0,110), \ldots, (9,110) \,\} \end{array} \\[2ex]
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \{\, (-1,100), \ldots, (9,100) \,\} \\[4ex]
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \{\, (-1,110) \,\}
\end{cases}
\qquad \text{iteration 6}
$$

Resolution (example)

iteration 7

$$\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\[2ex]
\mathcal{X}_2 = \mathsf{C}[\![\,\mathtt{X} := [0,10]\,]\!]\,\mathcal{X}_1 & [0,10] \times \mathbb{Z} \\[2ex]
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\,\mathtt{Y} := 100\,]\!]\,\mathcal{X}_2 \cup \\ \mathsf{C}[\![\,\mathtt{Y} := \mathtt{Y} + 10\,]\!]\,\mathcal{X}_5 \end{array} & \begin{array}{l} \{\,(0,100),\ldots,(10,100), \\ \ (-1,110),\ldots,(9,110)\,\} \end{array} \\[2ex]
\mathcal{X}_4 = \mathsf{C}[\![\,\mathtt{X} \geq 0\,]\!]\,\mathcal{X}_3 & \begin{array}{l} \{\,(0,100),\ldots,(10,100), \\ \ (0,110),\ldots,(9,110)\,\} \end{array} \\[2ex]
\mathcal{X}_5 = \mathsf{C}[\![\,\mathtt{X} := \mathtt{X} - 1\,]\!]\,\mathcal{X}_4 & \begin{array}{l} \{\,(-1,100),\ldots,(9,100), \\ \ (-1,110),\ldots,(8,110)\,\} \end{array} \\[2ex]
\mathcal{X}_6 = \mathsf{C}[\![\,\mathtt{X} < 0\,]\!]\,\mathcal{X}_3 & \{\,(-1,110)\,\}
\end{cases}$$

Resolution (example)

iteration 8

$$\begin{cases} \mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\ \mathcal{X}_2 = \mathsf{C}[\![\,\mathtt{X} := [0,10]\,]\!]\,\mathcal{X}_1 & [0,10] \times \mathbb{Z} \\ \mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\,\mathtt{Y} := 100\,]\!]\,\mathcal{X}_2 \cup \\ \mathsf{C}[\![\,\mathtt{Y} := \mathtt{Y} + 10\,]\!]\,\mathcal{X}_5 \end{array} & \begin{array}{l} \{\,(0,100),\ldots,(10,100), \\ \;(-1,110),\ldots,(9,110), \\ \;(-1,120),\ldots,(8,120)\,\} \end{array} \\ \mathcal{X}_4 = \mathsf{C}[\![\,\mathtt{X} \geq 0\,]\!]\,\mathcal{X}_3 & \begin{array}{l} \{\,(0,100),\ldots,(10,100), \\ \;(0,110),\ldots,(9,110)\,\} \end{array} \\ \mathcal{X}_5 = \mathsf{C}[\![\,\mathtt{X} := \mathtt{X} - 1\,]\!]\,\mathcal{X}_4 & \begin{array}{l} \{\,(-1,100),\ldots,(9,100), \\ \;(-1,110),\ldots,(8,110)\,\} \end{array} \\ \mathcal{X}_6 = \mathsf{C}[\![\,\mathtt{X} < 0\,]\!]\,\mathcal{X}_3 & \{\,(-1,110)\,\} \end{cases}$$

Resolution (example)

$$
\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 & \text{iteration 9} \\
& \mathbb{Z}^2 \\[1em]
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\[1em]
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \; (-1,110), \ldots, (9,110), \\ \; (-1,120), \ldots, (8,120) \,\} \end{array} \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \; (0,110), \ldots, (9,110), \\ \; (0,120), \ldots, (8,120) \,\} \end{array} \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \begin{array}{l} \{\, (-1,100), \ldots, (9,100), \\ \; (-1,110), \ldots, (8,110) \,\} \end{array} \\[1em]
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \{\, (-1,110), (-1,120) \,\}
\end{cases}
$$

Resolution (example)

$$
\left\{
\begin{array}{ll}
 & \text{iteration 10} \\
\mathcal{X}_1 = \mathbb{Z}^2 & \mathbb{Z}^2 \\[2ex]
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 & [0,10] \times \mathbb{Z} \\[2ex]
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \;(-1,110), \ldots, (9,110), \\ \;(-1,120), \ldots, (8,120) \,\} \end{array} \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 & \begin{array}{l} \{\, (0,100), \ldots, (10,100), \\ \;(0,110), \ldots, (9,110), \\ \;(0,120), \ldots, (8,120) \,\} \end{array} \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 & \begin{array}{l} \{\, (-1,100), \ldots, (9,100), \\ \;(-1,110), \ldots, (8,110), \\ \;(-1,120), \ldots, (7,120) \,\} \end{array} \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3 & \{\, (-1,110), (-1,120) \,\}
\end{array}
\right.
$$

Resolution (example)

$$
\begin{cases}
\mathcal{X}_1 = \mathbb{Z}^2 \\
\mathcal{X}_2 = \mathsf{C}[\![\, \mathtt{X} := [0,10] \,]\!]\, \mathcal{X}_1 \\
\mathcal{X}_3 = \begin{array}{l} \mathsf{C}[\![\, \mathtt{Y} := 100 \,]\!]\, \mathcal{X}_2 \cup \\ \mathsf{C}[\![\, \mathtt{Y} := \mathtt{Y} + 10 \,]\!]\, \mathcal{X}_5 \end{array} \\
\mathcal{X}_4 = \mathsf{C}[\![\, \mathtt{X} \geq 0 \,]\!]\, \mathcal{X}_3 \\
\mathcal{X}_5 = \mathsf{C}[\![\, \mathtt{X} := \mathtt{X} - 1 \,]\!]\, \mathcal{X}_4 \\
\mathcal{X}_6 = \mathsf{C}[\![\, \mathtt{X} < 0 \,]\!]\, \mathcal{X}_3
\end{cases}
$$

iteration ...

$\mathbb{Z}^2$

$[0,10] \times \mathbb{Z}$

$\{ (0,100), \ldots, (10,100), (-1,110), \ldots, (9,110), (-1,120), \ldots, (8,120), \ldots \}$

$\{ (0,100), \ldots, (10,100), (0,110), \ldots, (9,110), (0,120), \ldots, (8,120), \ldots \}$

$\{ (-1,100), \ldots, (9,100), (-1,110), \ldots, (8,110), (-1,120), \ldots, (7,120), \ldots \}$

$\{ (-1,110), (-1,120), \ldots \}$

Backward concrete semantics

Semantics of commands: $\overleftarrow{C}[\![\, c \,]\!] : \mathcal{P}(\mathbb{V} \to \mathbb{I}) \to \mathcal{P}(\mathbb{V} \to \mathbb{I})$

$$\overleftarrow{C}[\![\, \mathtt{V} := e \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{\, \rho \mid \exists v \in \mathrm{E}[\![\, e \,]\!]\, \rho,\ \rho[\, \mathtt{V} \mapsto v \,] \in \mathcal{X} \,\}$$
$$\overleftarrow{C}[\![\, e \bowtie 0 \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \mathrm{C}[\![\, e \bowtie 0 \,]\!]\, \mathcal{X}$$

(necessary conditions on ρ to have a successor in $\mathcal{X}$ by c)

Refinement decreasing iterations: given:

- a solution $(\mathcal{X}_\ell)_{\ell \in L}$ of the forward system
- an output criterion $\mathcal{Y}_x$

compute a least fixpoint by decreasing iterations [Bour93b]

$$\begin{cases} \mathcal{Y}_x^0 \stackrel{\text{def}}{=} \mathcal{X}_x \cap \mathcal{Y}_x \\ \mathcal{Y}_{\ell \neq x}^0 \stackrel{\text{def}}{=} \mathcal{X}_\ell \end{cases}$$
$$\begin{cases} \mathcal{Y}_x^{n+1} \stackrel{\text{def}}{=} \mathcal{X}_x \cap \mathcal{Y}_x \\ \mathcal{Y}_{\ell \neq x}^{n+1} \stackrel{\text{def}}{=} \mathcal{X}_\ell \cap \Big(\bigcup_{(\ell, c, \ell') \in A} \overleftarrow{C}[\![\, c \,]\!]\, \mathcal{Y}_{\ell'}^n \Big) \end{cases}$$

Limit to automation

We wish to perform automatic numerical invariant discovery.

Theoretical problems

- elements of $\mathcal{P}(\mathbb{V} \to \mathbb{I})$ are not computer representable
- transfer functions $\mathsf{C}[\![c]\!]$, $\overleftarrow{C}[\![c]\!]$ are not computable
- lattice iterations in $\mathcal{P}(\mathbb{V} \to \mathbb{I})$ are transfinite

Finding the best invariant is an undecidable problem

Note:

Even when $\mathbb{I}$ is finite, a concrete analysis is not tractable:

- representing elements in $\mathcal{P}(\mathbb{V} \to \mathbb{I})$ in extension is expensive
- computing $\mathsf{C}[\![c]\!]$, $\overleftarrow{C}[\![c]\!]$ explicitly is expensive
- the lattice $\mathcal{P}(\mathbb{V} \to \mathbb{I})$ has a large height ($\Rightarrow$ many iterations)

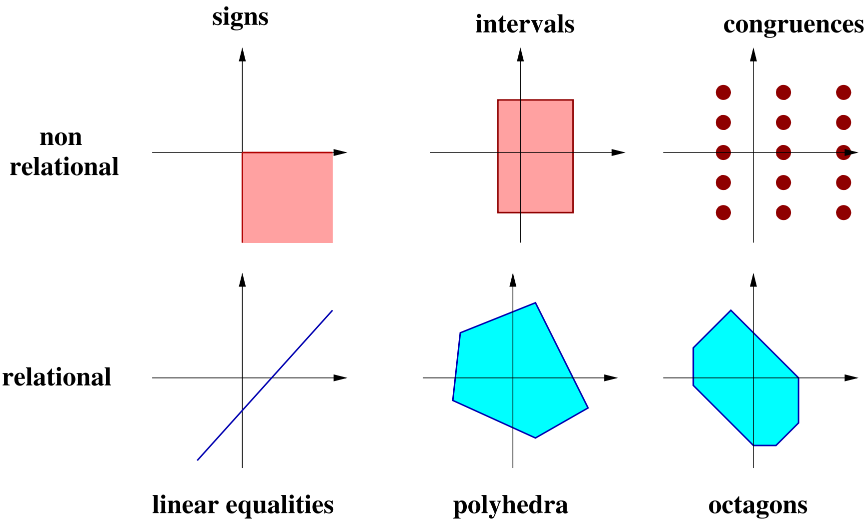
Abstraction

Numerical abstract domains

A numerical abstract domain is given by:

- a subset of $\mathcal{P}(\mathbb{V} \to \mathbb{I})$
  (a set of environment sets)
  together with a machine encoding,
- effective and sound abstract operators,
- an iteration strategy
  ensuring convergence in finite time.

Numerical abstract domain examples



Numerical abstract domains (cont.)

Representation: given by

- a set $\mathcal{D}^\sharp$ of machine-representable abstract values,
- a partial order $(\mathcal{D}^\sharp, \sqsubseteq, \bot^\sharp, \top^\sharp)$ relating the amount of information given by abstract values,
- a concretization function $\gamma \colon \mathcal{D}^\sharp \to \mathcal{P}(\mathbb{V} \to \mathbb{I})$ giving a concrete meaning to each abstract element.

Required algebraic properties:

- γ should be monotonic for $\sqsubseteq$: $\mathcal{X}^\sharp \sqsubseteq \mathcal{Y}^\sharp \implies \gamma(\mathcal{X}^\sharp) \subseteq \gamma(\mathcal{Y}^\sharp)$,
- $\gamma(\bot^\sharp) = \emptyset$,
- $\gamma(\top^\sharp) = \mathbb{V} \to \mathbb{I}$.

Note: γ need not be one-to-one.

Numerical abstract domains (cont.)

Abstract operators: we require:

- sound, effective, abstract transfer functions $C^\sharp[\![c]\!]$, $\overleftarrow{C}^\sharp[\![c]\!]$ for all commands c,
- sound, effective, abstract set operators $\cup^\sharp$, $\cap^\sharp$,
- an algorithm to decide the ordering $\sqsubseteq$.

Soundness criterion:

$F^\sharp$ is a sound abstraction of a $n-$ary operator F if:

$$\forall \mathcal{X}_1^\sharp, \ldots, \mathcal{X}_n^\sharp \in D^\sharp,\ F(\gamma(\mathcal{X}_1^\sharp), \ldots, \gamma(\mathcal{X}_n^\sharp)) \subseteq \gamma(F^\sharp(\mathcal{X}_1^\sharp, \ldots, \mathcal{X}_n^\sharp))$$

Both semantic and algorithmic aspects.

Abstract semantics

Abstract semantic equation system

$\mathcal{X}^\sharp : L \to \mathcal{D}^\sharp$

$$\mathcal{X}^\sharp_\ell \sqsupseteq \begin{cases} \mathcal{X}^\sharp_e & \text{if } \ell = e \qquad \text{(where } \mathcal{X}_e \subseteq \gamma(\mathcal{X}^\sharp_e)\text{)} \\ \bigcup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}^\sharp_{\ell'} & \text{if } \ell \neq e \qquad \text{(abstract transfer function)} \end{cases}$$

Soundness Theorem

Any solution $(\mathcal{X}^\sharp_\ell)_{\ell\in L}$ is a **sound over-approximation** of the concrete collecting semantics:

$$\forall \ell \in L,\ \gamma(\mathcal{X}^\sharp_\ell) \supseteq \mathcal{X}_\ell$$

where $\mathcal{X}_\ell$ is the smallest solution of

$$\begin{cases} \mathcal{X}_e & \text{given} \\ \mathcal{X}_\ell = \bigcup_{(\ell',c,\ell)\in A} \mathsf{C}[\![\, c \,]\!]\, \mathcal{X}_{\ell'} & \text{if } \ell \neq e \end{cases}$$

Iteration strategy

Resolution by iterations in $\mathcal{D}^\sharp$:

To effectively solve the abstract system, we require:

- an iteration ordering on abstract equations

  (which equation(s) are applied at a given iteration)

- a widening operator $\triangledown$ to speed-up the convergence, if there are infinite strictly increasing chains in $D^\sharp$.

  $\triangledown : (\mathcal{D}^\sharp \times \mathcal{D}^\sharp) \to \mathcal{D}^\sharp$ is a widening if:

  - it is sound: $\gamma(\mathcal{X}^\sharp) \cup \gamma(\mathcal{Y}^\sharp) \subseteq \gamma(\mathcal{X}^\sharp \triangledown \mathcal{Y}^\sharp)$
  - it enforces termination:

    $\forall$ sequence $(\mathcal{Y}_i^\sharp)_{i \in \mathbb{N}}$

    the sequence $\mathcal{X}_0^\sharp = \mathcal{Y}_0^\sharp$, $\mathcal{X}_{i+1}^\sharp = \mathcal{X}_i^\sharp \triangledown \mathcal{Y}_{i+1}^\sharp$

    stabilizes in finite time: $\exists n < \omega$, $\mathcal{X}_{n+1}^\sharp = \mathcal{X}_n^\sharp$

    (note: $\exists n$, $\forall m \geq n$, $\mathcal{X}_{m+1}^\sharp = \mathcal{X}_m^\sharp$ is not required)

Abstract analysis

$\mathcal{W} \subseteq L$ is a set of widening points if every CFG cycle has a point in $\mathcal{W}$.

Forward analysis:

$$\mathcal{X}_e^{\sharp 0} \stackrel{\text{def}}{=} \mathcal{X}_e^\sharp \quad \text{given, such that } \mathcal{X}_e \subseteq \gamma(\mathcal{X}_e^\sharp)$$

$$\mathcal{X}_{\ell \neq e}^{\sharp 0} \stackrel{\text{def}}{=} \bot^\sharp$$

$$\mathcal{X}_\ell^{\sharp n+1} \stackrel{\text{def}}{=} \begin{cases} \mathcal{X}_e^\sharp & \text{if } \ell = e \\ \bigcup^\sharp_{(\ell', c, \ell) \in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp n} & \text{if } \ell \notin \mathcal{W},\, \ell \neq e \\ \mathcal{X}_\ell^{\sharp n} \,\triangledown\, \bigcup^\sharp_{(\ell', c, \ell) \in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp n} & \text{if } \ell \in \mathcal{W},\, \ell \neq e \end{cases}$$

- termination: for some δ, $\forall \ell,\ \mathcal{X}_\ell^{\sharp \delta+1} = \mathcal{X}_\ell^{\sharp \delta}$
- soundness: $\forall \ell \in L,\ \mathcal{X}_\ell \subseteq \gamma(\mathcal{X}_\ell^{\sharp \delta})$
- can be refined by decreasing iterations with narrowing $\vartriangle$ (presented later)
- here, apply every equation at each step, but other iteration scheme are possible (worklist, chaotic iterations, see [Bour93a])

Abstract analysis (proof)

<u>Proof of soundness:</u>

Suppose that $\forall \ell$, $\mathcal{X}_\ell^{\sharp\delta+1} = \mathcal{X}_\ell^{\sharp\delta}$.

If $\ell = e$, by definition: $\mathcal{X}_e^{\sharp\delta} = \mathcal{X}_e^\sharp$ and $\mathcal{X}_e \subseteq \gamma(\mathcal{X}_e^{\sharp\delta})$.
If $\ell \neq e$, $\ell \notin \mathcal{W}$, then $\mathcal{X}_\ell^{\sharp\delta} = \mathcal{X}_\ell^{\sharp\delta+1} = \cup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp\,\delta}$.
By soundness of $\cup^\sharp$ and $\mathsf{C}^\sharp[\![\, c \,]\!]$, $\gamma(\mathcal{X}_\ell^{\sharp\delta}) \supseteq \cup_{(\ell',c,\ell)\in A} \mathsf{C}[\![\, c \,]\!]\, \gamma(\mathcal{X}_{\ell'}^{\sharp\,\delta})$.
If $\ell \neq e$, $\ell \in \mathcal{W}$, then $\mathcal{X}_\ell^{\sharp\delta} = \mathcal{X}_\ell^{\sharp\delta+1} = \mathcal{X}_\ell^{\sharp\delta} \,\triangledown\, \cup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp\,\delta}$.
By soundness of $\triangledown$, $\gamma(\mathcal{X}_\ell^{\sharp\delta}) \supseteq \gamma(\cup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp\,\delta})$,
and so we also have $\gamma(\mathcal{X}_\ell^{\sharp\delta}) \supseteq \cup_{(\ell',c,\ell)\in A} \mathsf{C}[\![\, c \,]\!]\, \gamma(\mathcal{X}_{\ell'}^{\sharp\,\delta})$.

We have proved that $\lambda\ell.\gamma(\mathcal{X}_\ell^{\sharp\delta})$ is a postfixpoint of the concrete equation system.
Hence, it is greater than its least solution.

Abstract analysis (proof)

Proof of termination:

Suppose that the iteration does not terminate in finite time.

Given a label $\ell \in L$, we denote by $i_\ell^1, \ldots, i_\ell^k, \ldots$ the increasing sequence of unstable indices, i.e., such that $\forall k$, $\mathcal{X}^{\sharp}{}_\ell^{i_\ell^k+1} \neq \mathcal{X}_\ell^{\sharp\, i_\ell^k}$.

As the iteration is not stable, $\forall n$, $\exists \ell$, $\mathcal{X}_\ell^{\sharp\, n} \neq \mathcal{X}_\ell^{\sharp\, n+1}$.

Hence, the sequence $(i_\ell^k)_k$ is infinite for at least one $\ell \in L$.

We argue that $\exists \ell \in \mathcal{W}$ such that $(i_\ell^k)_k$ is infinite as, otherwise,
$N = \max\{\, i_\ell^k \mid \ell \in \mathcal{W}\,\} + |L|$ is finite and satisfies: $\forall n \geq N, \forall \ell \in L$, $\mathcal{X}_\ell^{\sharp\, n} = \mathcal{X}_\ell^{\sharp\, n+1}$,
contradicting our assumption.

For such a $\ell \in \mathcal{W}$, consider the subsequence $\mathcal{Y}_k^\sharp = \mathcal{X}_\ell^{\sharp\, i_\ell^k}$ comprised of the unstable iterates of $\mathcal{X}_\ell^\sharp$.

Then $\mathcal{Y}^{\sharp\, k+1} = \mathcal{Y}^{\sharp\, k} \triangledown \mathcal{Z}^{\sharp\, k}$ for some sequence $\mathcal{Z}^{\sharp\, k}$.

The subsequence is infinite and $\forall k$, $\mathcal{Y}^{\sharp\, k+1} \neq \mathcal{Y}^{\sharp\, k}$, which contradicts the definition of $\triangledown$.

Hence, the iteration must terminate in finite time.

Abstract analysis (cont.)

Backward refinement:

Given a forward analysis result $\mathcal{X}^\sharp$ and an abstract output $\mathcal{Y}^\sharp_x$.

$$\mathcal{Y}^{\sharp 0}_x \stackrel{\text{def}}{=} \mathcal{X}^\sharp_x \cap^\sharp \mathcal{Y}^\sharp_x$$

$$\mathcal{Y}^{\sharp 0}_{\ell \neq x} \stackrel{\text{def}}{=} \mathcal{X}^\sharp_\ell$$

$$\mathcal{Y}^{\sharp n+1}_\ell \stackrel{\text{def}}{=} \begin{cases} \mathcal{X}^\sharp_x \cap^\sharp \mathcal{Y}^\sharp_x & \text{if } \ell = x \\ \mathcal{X}^\sharp_\ell \cap^\sharp \bigcup^\sharp_{(\ell,c,\ell') \in A} \overleftarrow{C}^\sharp [\![c]\!] \mathcal{Y}^{\sharp n}_{\ell'} & \text{if } \ell \notin \mathcal{W},\ \ell \neq x \\ \mathcal{Y}^{\sharp n}_\ell \bigtriangleup (\mathcal{X}^\sharp_\ell \cap^\sharp \bigcup^\sharp_{(\ell,c,\ell') \in A} \overleftarrow{C}^\sharp [\![c]\!] \mathcal{Y}^{\sharp n}_{\ell'}) & \text{if } \ell \in \mathcal{W},\ \ell \neq x \end{cases}$$

$\bigtriangleup$ overapproximates $\cap$ while enforcing the convergence of decreasing iterations (the definition will be given later, on intervals)

Forward–backward analyses can be iterated [Bour93b].

Exact and best abstractions: Reminders

Galois connection: $(\mathcal{D}, \subseteq) \xleftrightarrows[\alpha]{\gamma} (\mathcal{D}^\sharp, \sqsubseteq)$

- α, γ monotonic and $\forall \mathcal{X}, \mathcal{Y}^\sharp$, $\alpha(\mathcal{X}) \sqsubseteq \mathcal{Y}^\sharp \iff \mathcal{X} \subseteq \gamma(\mathcal{Y}^\sharp)$
- $\Rightarrow$ elements $\mathcal{X}$ have a best abstraction: $\alpha(\mathcal{X})$
- $\Rightarrow$ operators F have a best abstraction: $F^\sharp = \alpha \circ F \circ \gamma$

Sometimes, no α exists:

- $\{\, \gamma(\mathcal{Y}^\sharp) \,|\, \mathcal{X} \subseteq \gamma(\mathcal{Y}^\sharp) \,\}$ has no greatest lower bound
- abstract elements with the same γ have no best representation

$\alpha \circ F \circ \gamma$ may still be defined for some F (partial α)

Concretization-based optimality:

- sound abstraction: $\gamma \circ F^\sharp \supseteq F \circ \gamma$
- exact abstraction: $\gamma \circ F^\sharp = F \circ \gamma$
- optimal abstraction: $\gamma(\mathcal{X}^\sharp)$ minimal in $\{\, \gamma(\mathcal{Y}^\sharp) \,|\, \mathcal{X} \subseteq \gamma(\mathcal{Y}^\sharp) \,\}$

Non-relational domains

Value abstract domain

Idea: start from an abstraction of values $\mathcal{P}(\mathbb{I})$

Numerical value abstract domain:

$\mathcal{B}^\sharp$	abstract values, machine-representable
γ_b: $\mathcal{B}^\sharp \to \mathcal{P}(\mathbb{I})$	concretization
$\sqsubseteq_b$	partial order
$\bot_b^\sharp$, $\top_b^\sharp$	represent $\emptyset$ and $\mathbb{I}$
$\cup_b^\sharp$, $\cap_b^\sharp$	abstractions of $\cup$ and $\cap$
$\triangledown_b$	extrapolation operator
α_b: $\mathcal{P}(\mathbb{I}) \to \mathcal{B}^\sharp$	abstraction (optional)

Derived abstract domain

$$\mathcal{D}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \to (\mathcal{B}^\sharp \setminus \{\,\perp_b^\sharp\,\})) \cup \{\,\perp^\sharp\,\}$$

- point-wise extension: $\mathcal{X}^\sharp \in \mathcal{D}^\sharp$ is a vector of elements in $\mathcal{B}^\sharp$
  (e.g. using arrays of size $|\mathbb{V}|$)
- smashed $\perp^\sharp$ (avoids redundant representations of $\emptyset$)

Definitions on $\mathcal{D}^\sharp$ derived from $\mathcal{B}^\sharp$:

$$\gamma(\mathcal{X}^\sharp) \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{if } \mathcal{X}^\sharp = \perp^\sharp \\ \{\,\rho \mid \forall \mathtt{V},\ \rho(\mathtt{V}) \in \gamma_b(\mathcal{X}^\sharp(\mathtt{V}))\,\} & \text{otherwise} \end{cases}$$

$$\alpha(\mathcal{X}) \stackrel{\text{def}}{=} \begin{cases} \perp^\sharp & \text{if } \mathcal{X} = \emptyset \\ \lambda \mathtt{V}.\alpha_b(\{\,\rho(\mathtt{V}) \mid \rho \in \mathcal{X}\,\}) & \text{otherwise} \end{cases}$$

$$\top^\sharp \stackrel{\text{def}}{=} \lambda \mathtt{V}.\top_b^\sharp$$

Derived abstract domain (cont.)

$$\mathcal{X}^\sharp \sqsubseteq \mathcal{Y}^\sharp \overset{\text{def}}{\iff} \mathcal{X}^\sharp = \bot^\sharp \vee (\mathcal{X}^\sharp, \mathcal{Y}^\sharp \neq \bot^\sharp \wedge \forall \mathtt{V},\, \mathcal{X}^\sharp(\mathtt{V}) \sqsubseteq_b \mathcal{Y}^\sharp(\mathtt{V}))$$

$$\mathcal{X}^\sharp \cup^\sharp \mathcal{Y}^\sharp \overset{\text{def}}{=} \begin{cases} \mathcal{Y}^\sharp & \text{if } \mathcal{X}^\sharp = \bot^\sharp \\ \mathcal{X}^\sharp & \text{if } \mathcal{Y}^\sharp = \bot^\sharp \\ \lambda \mathtt{V}.\mathcal{X}^\sharp(\mathtt{V}) \cup_b^\sharp \mathcal{Y}^\sharp(\mathtt{V}) & \text{otherwise} \end{cases}$$

$$\mathcal{X}^\sharp \triangledown \mathcal{Y}^\sharp \overset{\text{def}}{=} \begin{cases} \mathcal{Y}^\sharp & \text{if } \mathcal{X}^\sharp = \bot^\sharp \\ \mathcal{X}^\sharp & \text{if } \mathcal{Y}^\sharp = \bot^\sharp \\ \lambda \mathtt{V}.\mathcal{X}^\sharp(\mathtt{V}) \triangledown_b \mathcal{Y}^\sharp(\mathtt{V}) & \text{otherwise} \end{cases}$$

$$\mathcal{X}^\sharp \cap^\sharp \mathcal{Y}^\sharp \overset{\text{def}}{=} \begin{cases} \bot^\sharp & \text{if } \mathcal{X}^\sharp = \bot^\sharp \text{ or } \mathcal{Y}^\sharp = \bot^\sharp \\ \bot^\sharp & \text{if } \exists \mathtt{V},\, \mathcal{X}^\sharp(\mathtt{V}) \cap_b^\sharp \mathcal{Y}^\sharp(\mathtt{V}) = \bot_b^\sharp \\ \lambda \mathtt{V}.\mathcal{X}^\sharp(\mathtt{V}) \cap_b^\sharp \mathcal{Y}^\sharp(\mathtt{V}) & \text{otherwise} \end{cases}$$

We will see later how to derive $\mathsf{C}^\sharp[\![\, c \,]\!]$, $\overleftarrow{C}^\sharp[\![\, c \,]\!]$ using:

- abstract operators $+_b^\sharp$, … for $\mathsf{C}^\sharp[\![\, \mathtt{V} := e \,]\!]$
- backward abstract operators $\overleftarrow{+}_b^\sharp$, …
  for $\overleftarrow{C}^\sharp[\![\, \mathtt{V} := e \,]\!]$ and $\mathsf{C}^\sharp[\![\, e \bowtie 0 \,]\!]^\sharp$

Cartesian abstraction

Non-relational domains "forget" all relationships between variables.

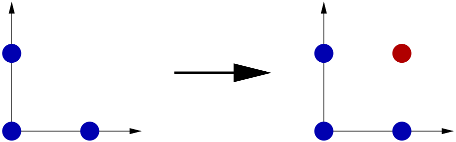
Cartesian abstraction:

Upper closure operator $\rho_c : \mathcal{P}(\mathbb{V} \to \mathbb{I}) \to \mathcal{P}(\mathbb{V} \to \mathbb{I})$

$$\rho_c(\mathcal{X}) \stackrel{\text{def}}{=} \{\ \rho \in \mathbb{V} \to \mathbb{I} \mid \forall \mathtt{V} \in \mathbb{V},\ \exists \rho' \in \mathcal{X},\ \rho(\mathtt{V}) = \rho'(\mathtt{V})\ \}$$

A domain is non relational if $\rho \circ \gamma = \gamma$,
i.e. it cannot distinguish between $\mathcal{X}$ and $\mathcal{X}'$ if $\rho_c(\mathcal{X}) = \rho_c(\mathcal{X}')$.

Example: $\rho_c(\{(X, Y) \mid X \in \{0, 2\}, Y \in \{0, 2\},\ X + Y \leq 2\}) = \{0, 2\} \times \{0, 2\}$.



Data-structures for non-relational domains

Arrays

- $\mathcal{O}(1)$ to read or modify a variable
- $\mathcal{O}(|\mathbb{V}|)$ for a copy or a binary operator ($\cup^\sharp$, $\cap^\sharp$, etc.)

Functional arrays e.g.: balanced binary trees

- $\mathcal{O}(\log|\mathbb{V}|)$ to read or modify a variable
- $\mathcal{O}(1)$ to copy
- $\mathcal{O}(|\mathcal{X}^\sharp \Delta \mathcal{Y}^\sharp| \times \log|\mathbb{V}|)$ for a binary operator $\mathcal{X}^\sharp \cup^\sharp \mathcal{Y}^\sharp$, etc.
  (Δ is the symmetric difference)

In practice, $|\mathcal{X}^\sharp \Delta \mathcal{Y}^\sharp| \ll |\mathbb{V}|$.

Generic non-relational abstract assignments

Given: sound abstract versions in $\mathcal{B}^\sharp$ of all arithmetic operators:

$$
\begin{array}{rccc}
[c,c']^\sharp_b : & \{\, x \mid c \leq x \leq c' \,\} & \subseteq & \gamma_b([c,c']^\sharp_b) \\
-^\sharp_b : & \{\, -x \mid x \in \gamma_b(\mathcal{X}^\sharp_b) \,\} & \subseteq & \gamma_b(-^\sharp_b\ \mathcal{X}^\sharp_b) \\
+^\sharp_b : & \{\, x+y \mid x \in \gamma_b(\mathcal{X}^\sharp_b), y \in \gamma_b(\mathcal{Y}^\sharp_b) \,\} & \subseteq & \gamma_b(\mathcal{X}^\sharp_b +^\sharp_b \mathcal{Y}^\sharp_b) \\
\vdots & & &
\end{array}
$$

We can define:

- an abstract semantics of expressions: $\mathsf{E}^\sharp[\![\, e \,]\!] : \mathcal{D}^\sharp \to \mathcal{B}^\sharp$

$$
\begin{array}{lcl}
\mathsf{E}^\sharp[\![\, e \,]\!]\, \bot^\sharp & \stackrel{\text{def}}{=} & \bot^\sharp_b \\
\text{if } \mathcal{X}^\sharp \neq \bot^\sharp : & & \\
\mathsf{E}^\sharp[\![\, [c,c'] \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & [c,c']^\sharp_b \\
\mathsf{E}^\sharp[\![\, \mathtt{V} \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & \mathcal{X}^\sharp(\mathtt{V}) \\
\mathsf{E}^\sharp[\![\, -e \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & -^\sharp_b\ \mathsf{E}^\sharp[\![\, e \,]\!]\, \mathcal{X}^\sharp \\
\mathsf{E}^\sharp[\![\, e_1 + e_2 \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & \mathsf{E}^\sharp[\![\, e_1 \,]\!]\, \mathcal{X}^\sharp +^\sharp_b \mathsf{E}^\sharp[\![\, e_2 \,]\!]\, \mathcal{X}^\sharp \\
\vdots & &
\end{array}
$$

Generic non-relational abstract assignments (cont.)

We can then define:

- an abstract assignment:

$$C^\sharp[\![\, \mathtt{V} := e \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot^\sharp & \text{if } \mathcal{V}_b^\sharp = \bot_b^\sharp \\ \mathcal{X}^\sharp[\mathtt{V} \mapsto \mathcal{V}_b^\sharp] & \text{otherwise} \end{cases}$$

where $\mathcal{V}_b^\sharp = E^\sharp[\![\, e \,]\!]\, \mathcal{X}^\sharp$.

Using a Galois connection (α_b, γ_b):

We can define best abstract arithmetic operators:

$$\begin{array}{lcl} [c, c']_b^\sharp & \stackrel{\text{def}}{=} & \alpha_b(\{\, x \mid c \leq x \leq c' \,\}) \\ -_b^\sharp\, \mathcal{X}_b^\sharp & \stackrel{\text{def}}{=} & \alpha_b(\{\, -x \mid x \in \gamma(\mathcal{X}_b^\sharp) \,\}) \\ \mathcal{X}_b^\sharp +_b^\sharp \mathcal{Y}_b^\sharp & \stackrel{\text{def}}{=} & \alpha_b(\{\, x+y \mid x \in \gamma(\mathcal{X}_b^\sharp),\ y \in \gamma(\mathcal{Y}_b^\sharp) \,\}) \\ \vdots & & \end{array}$$

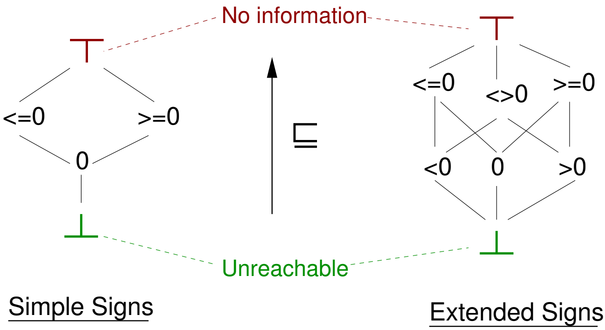
Note: in general, $E^\sharp[\![\, e \,]\!]$ is less precise than $\alpha_b \circ E[\![\, e \,]\!] \circ \gamma$

e.g. $e = \mathtt{V} - \mathtt{V}$ and $\gamma_b(\mathcal{X}^\sharp(\mathtt{V})) = [0, 1]$

The sign domain

The sign lattices

Hasse diagram: for the lattice $(\mathcal{B}^\sharp, \sqsubseteq_b, \perp_b^\sharp, \top_b^\sharp)$



The extended sign domain is a refinement of the simple sign domain.

The diagram implicitly defines $\cup^\sharp$ and $\cap^\sharp$ as the least upper bound and greatest lower bound for $\sqsubseteq$.

Operations on simple signs

Abstraction α: there is a Galois connection between $\mathcal{B}^\sharp$ and $\mathcal{P}(\mathbb{I})$:

$$\alpha_b(S) \stackrel{\text{def}}{=} \begin{cases} \perp_b^\sharp & \text{if } S = \emptyset \\ 0 & \text{if } S = \{0\} \\ \geq 0 & \text{else if } \forall s \in S,\ s \geq 0 \\ \leq 0 & \text{else if } \forall s \in S,\ s \leq 0 \\ \top_b^\sharp & \text{otherwise} \end{cases}$$

Derived abstract arithmetic operators:

$$c_b^\sharp \stackrel{\text{def}}{=} \alpha_b(\{c\}) = \begin{cases} 0 & \text{if } c = 0 \\ \leq 0 & \text{if } c < 0 \\ \geq 0 & \text{if } c > 0 \end{cases}$$

$$\begin{aligned} X^\sharp +_b^\sharp Y^\sharp &\stackrel{\text{def}}{=} \alpha_b(\{\, x + y \mid x \in \gamma_b(X^\sharp),\ y \in \gamma_b(Y^\sharp) \,\}) \\ &= \begin{cases} \perp_b^\sharp & \text{if } X \text{ or } Y^\sharp = \perp_b^\sharp \\ 0 & \text{if } X^\sharp = Y^\sharp = 0 \\ \leq 0 & \text{else if } X^\sharp \text{ and } Y^\sharp \in \{0, \leq 0\} \\ \geq 0 & \text{else if } X^\sharp \text{ and } Y^\sharp \in \{0, \geq 0\} \\ \top_b^\sharp & \text{otherwise} \end{cases} \end{aligned}$$

Operations on simple signs (cont.)

Abstract test examples:

$$\mathsf{C}^\sharp[\![\, \mathtt{X} \leq 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \left(\begin{cases} \mathcal{X}^\sharp[\mathtt{X} \mapsto 0] & \text{if } \mathcal{X}^\sharp(\mathtt{X}) \in \{0, \geq 0\} \\ \mathcal{X}^\sharp[\mathtt{X} \mapsto \leq 0] & \text{if } \mathcal{X}^\sharp(\mathtt{X}) \in \{\top_b^\sharp, \leq 0\} \\ \bot^\sharp & \text{otherwise} \end{cases} \right)$$

$$\mathsf{C}^\sharp[\![\, \mathtt{X} - c \leq 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \left(\begin{cases} \mathsf{C}^\sharp[\![\, \mathtt{X} \leq 0 \,]\!]\, \mathcal{X}^\sharp & \text{if } c \leq 0 \\ \mathcal{X}^\sharp & \text{otherwise} \end{cases} \right)$$

$$\mathsf{C}^\sharp[\![\, \mathtt{X} - \mathtt{Y} \leq 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=}$$
$$\begin{cases} \mathsf{C}^\sharp[\![\, \mathtt{X} \leq 0 \,]\!]\, \mathcal{X}^\sharp & \text{if } \mathcal{X}^\sharp(\mathtt{Y}) \in \{0, \leq 0\} \\ \mathcal{X}^\sharp & \text{otherwise} \end{cases} \quad \cap^\sharp$$
$$\begin{cases} \mathsf{C}^\sharp[\![\, \mathtt{Y} \geq 0 \,]\!]\, \mathcal{X}^\sharp & \text{if } \mathcal{X}^\sharp(\mathtt{X}) \in \{0, \geq 0\} \\ \mathcal{X}^\sharp & \text{otherwise} \end{cases}$$

Other cases: $\mathsf{C}^\sharp[\![\, \mathit{expr} \bowtie 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \mathcal{X}^\sharp$ is always a sound abstraction.

Simple sign analysis example

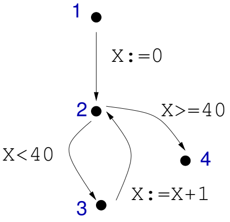
Example analysis using the simple sign domain:

```
X:=0;
while X<40 do
  X:=X+1
done
```

Program

$$\begin{cases} \mathcal{X}_2^{\sharp i+1} = \mathsf{C}^\sharp[\![\, \mathtt{X} := 0 \,]\!]\, \mathcal{X}_1^{\sharp i} \cup \\ \qquad\qquad \mathsf{C}^\sharp[\![\, \mathtt{X} := \mathtt{X} + 1 \,]\!]\, \mathcal{X}_3^{\sharp i} \\ \mathcal{X}_3^{\sharp i+1} = \mathsf{C}^\sharp[\![\, \mathtt{X} < 40 \,]\!]\, \mathcal{X}_2^{\sharp i} \\ \mathcal{X}_4^{\sharp i+1} = \mathsf{C}^\sharp[\![\, \mathtt{X} \geq 40 \,]\!]\, \mathcal{X}_2^{\sharp i} \end{cases}$$

Iteration system



CFG

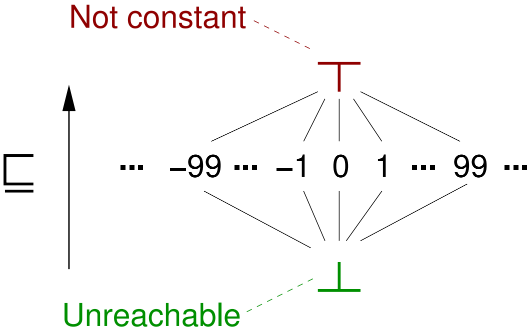
ℓ	$\mathcal{X}_\ell^{\sharp 0}$	$\mathcal{X}_\ell^{\sharp 1}$	$\mathcal{X}_\ell^{\sharp 2}$	$\mathcal{X}_\ell^{\sharp 3}$	$\mathcal{X}_\ell^{\sharp 4}$	$\mathcal{X}_\ell^{\sharp 5}$
1	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$
2	$\bot^\sharp$	$\mathtt{X} = 0$	$\mathtt{X} = 0$	$\mathtt{X} \geq 0$	$\mathtt{X} \geq 0$	$\mathtt{X} \geq 0$
3	$\bot^\sharp$	$\bot^\sharp$	$\mathtt{X} = 0$	$\mathtt{X} = 0$	$\mathtt{X} \geq 0$	$\mathtt{X} \geq 0$
4	$\bot^\sharp$	$\bot^\sharp$	$\mathtt{X} = 0$	$\mathtt{X} = 0$	$\mathtt{X} \geq 0$	$\mathtt{X} \geq 0$

Iterations

The constant domain

The constant lattice

Hasse diagram:



$\mathcal{B}^\sharp = \mathbb{I} \cup \{\top_b^\sharp; \perp_b^\sharp\}$

The lattice is flat but infinite.

Operations on constants

Abstraction α: there is a Galois connection:

$$\alpha_b(S) \stackrel{\text{def}}{=} \begin{cases} \perp_b^\sharp & \text{if } S = \emptyset \\ c & \text{if } S = \{c\} \\ \top_b^\sharp & \text{otherwise} \end{cases}$$

Derived abstract arithmetic operators:

$$c_b^\sharp \stackrel{\text{def}}{=} c$$

$$(X^\sharp) +_b^\sharp (Y^\sharp) \stackrel{\text{def}}{=} \begin{cases} \perp_b^\sharp & \text{if } X^\sharp \text{ or } Y^\sharp = \perp_b^\sharp \\ \top_b^\sharp & \text{else if } X^\sharp \text{ or } Y^\sharp = \top_b^\sharp \\ X^\sharp + Y^\sharp & \text{otherwise} \end{cases}$$

$$(X^\sharp) \times_b^\sharp (Y^\sharp) \stackrel{\text{def}}{=} \begin{cases} \perp_b^\sharp & \text{if } X^\sharp \text{ or } Y^\sharp = \perp_b^\sharp \\ 0 & \text{else if } X^\sharp \text{ or } Y^\sharp = 0 \\ \top_b^\sharp & \text{else if } X^\sharp \text{ or } Y^\sharp = \top_b^\sharp \\ X^\sharp \times Y^\sharp & \text{otherwise} \end{cases}$$

Operations on constants (cont.)

Abstract test examples:

$$\mathsf{C}^\sharp[\![\, \mathtt{X} - c = 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot^\sharp & \text{if } \mathcal{X}^\sharp(\mathtt{X}) \notin \{c, \top_b^\sharp\} \\ \mathcal{X}^\sharp[\mathtt{X} \mapsto c] & \text{otherwise} \end{cases}$$

$$\mathsf{C}^\sharp[\![\, \mathtt{X} - \mathtt{Y} - c = 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=}$$

$$\left(\begin{cases} \mathsf{C}^\sharp[\![\, \mathtt{X} - (\mathcal{X}^\sharp(\mathtt{Y}) + c) = 0 \,]\!]\, \mathcal{X}^\sharp & \text{if } \mathcal{X}^\sharp(\mathtt{Y}) \notin \{\bot_b^\sharp, \top_b^\sharp\} \\ \mathcal{X}^\sharp & \text{otherwise} \end{cases} \right) \cap^\sharp$$
$$\left(\begin{cases} \mathsf{C}^\sharp[\![\, \mathtt{Y} - (\mathcal{X}^\sharp(\mathtt{X}) - c) = 0 \,]\!]\, \mathcal{X}^\sharp & \text{if } \mathcal{X}^\sharp(\mathtt{X}) \notin \{\bot_b^\sharp, \top_b^\sharp\} \\ \mathcal{X}^\sharp & \text{otherwise} \end{cases} \right)$$

Constant analysis example

$\mathcal{B}^\sharp$ has finite height, the $(\mathcal{X}_\ell^{\sharp i})$ converge in finite time.

(even though $\mathcal{B}^\sharp$ is infinite...)

Analysis example:

```
X:=0; Y:=10;
while X<100 do
  Y:=Y-3;
  X:=X+Y; •
  Y:=Y+3
done
```

The constant analysis finds, at •, the invariant: $\begin{cases} \mathtt{X} = \top_b^\sharp \\ \mathtt{Y} = 7 \end{cases}$

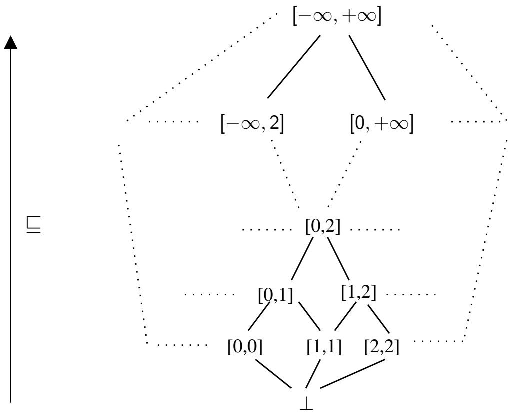
Note: the analysis can find constants that do not appear syntactically in the program.

The interval domain

The interval lattice

Introduced by [Cous76].

$$\mathcal{B}^\sharp \stackrel{\text{def}}{=} \{\, [a,b] \mid a \in \mathbb{I} \cup \{\,-\infty\,\},\ b \in \mathbb{I} \cup \{\,+\infty\,\},\ a \leq b \,\} \ \cup\ \{\, \perp_b^\sharp \,\}$$



<u>Note:</u> intervals are open at infinite bounds $+\infty$, $-\infty$.

The interval lattice (cont.)

Galois connection (α_b, γ_b):

$$\gamma_b([a,b]) \stackrel{\text{def}}{=} \{\, x \in \mathbb{I} \mid a \leq x \leq b \,\}$$

$$\alpha_b(\mathcal{X}) \stackrel{\text{def}}{=} \begin{cases} \perp_b^\sharp & \text{if } \mathcal{X} = \emptyset \\ [\min \mathcal{X}, \max \mathcal{X}] & \text{otherwise} \end{cases}$$

If $\mathbb{I} = \mathbb{Q}$, α_b is not always defined...

Partial order:

$$[a,b] \sqsubseteq_b [c,d] \stackrel{\text{def}}{\iff} a \geq c \text{ and } b \leq d$$

$$\top_b^\sharp \stackrel{\text{def}}{=}]-\infty, +\infty[$$

$$[a,b] \cup_b^\sharp [c,d] \stackrel{\text{def}}{=} [\min(a,c), \max(b,d)]$$

$$[a,b] \cap_b^\sharp [c,d] \stackrel{\text{def}}{=} \begin{cases} [\max(a,c), \min(b,d)] & \text{if } \max \leq \min \\ \perp_b^\sharp & \text{otherwise} \end{cases}$$

If $\mathbb{I} \neq \mathbb{Q}$, it is a complete lattice.

Interval abstract arithmetic operators

$$
\begin{array}{lcl}
[c,c']^\sharp_b & \stackrel{\text{def}}{=} & [c,c'] \\
-^\sharp_b\,[a,b] & \stackrel{\text{def}}{=} & [-b,-a] \\
[a,b] +^\sharp_b [c,d] & \stackrel{\text{def}}{=} & [a+c,b+d] \\
[a,b] -^\sharp_b [c,d] & \stackrel{\text{def}}{=} & [a-d,b-c] \\
[a,b] \times^\sharp_b [c,d] & \stackrel{\text{def}}{=} & [\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)] \\
[a,b] \,/^\sharp_b\, [c,d] & \stackrel{\text{def}}{=} & \begin{cases} \bot^\sharp_b & \text{if } c=d=0 \\ [\min(a/c,a/d,b/c,b/d), & \text{else if } 0 \le c \\ \ \max(a/c,a/d,b/c,b/d)] & \\ [-b,-a]/^\sharp_b[-d,-c] & \text{else if } d \le 0 \\ ([a,b]/^\sharp_b[c,0]) \cup^\sharp_b ([a,b]/^\sharp_b[0,d]) & \text{otherwise} \end{cases}
\end{array}
$$

where $\left|\begin{array}{l} \pm\infty \times 0 = 0, \quad 0/0 = 0, \quad \forall x,\, x/\pm\infty = 0 \\ \forall x > 0,\, x/0 = +\infty, \quad \forall x < 0,\, x/0 = -\infty \end{array}\right.$

Operators are strict: $-^\sharp_b \bot^\sharp_b = \bot^\sharp_b$, $[a,b] +^\sharp_b \bot^\sharp_b = \bot^\sharp_b$, etc.

Exactness and optimality: Example proofs

Proof: exactness of $+^\sharp_b$

$$\begin{array}{ll} & \{\, x+y \mid x \in \gamma_b([a,b]),\ y \in \gamma_b([c,d]) \,\} \\ = & \{\, x+y \mid a \leq x \leq b \wedge c \leq y \leq d \,\} \\ = & \{\, z \mid a+c \leq z \leq b+d \,\} \\ = & \gamma_b([a+c, b+d]) \\ = & \gamma_b([a,b] \ +^\sharp_b \ [c,d]) \end{array}$$

Proof optimality of $\cup^\sharp_b$

$$\begin{array}{ll} & \alpha_b(\gamma_b([a,b]) \cup \gamma_b([c,d])) \\ = & \alpha_b(\{\, x \mid a \leq x \leq b \,\} \cup \{\, x \mid c \leq x \leq d \,\}) \\ = & \alpha_b(\{\, x \mid a \leq x \leq b \vee c \leq x \leq d \,\}) \\ = & [\min \{\, x \mid a \leq x \leq b \vee c \leq x \leq d \,\}, \max \{\, x \mid a \leq x \leq b \vee c \leq x \leq d \,\}] \\ = & [\min(a,c), \max(b,d)] \\ = & [a,b] \cup^\sharp_b [c,d] \end{array}$$

but $\cup^\sharp_b$ is not exact

...

Interval abstract tests (non-generic)

If $\mathcal{X}^\sharp(\mathtt{X}) = [a, b]$ and $\mathcal{X}^\sharp(\mathtt{Y}) = [c, d]$, we can define:

$$C^\sharp[\![\, \mathtt{X} - c \leq 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot^\sharp & \text{if } a > c \\ \mathcal{X}^\sharp[\, \mathtt{X} \mapsto [a, \min(b, c)] \,] & \text{otherwise} \end{cases}$$

$$C^\sharp[\![\, \mathtt{X} - \mathtt{Y} \leq 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot^\sharp & \text{if } a > d \\ \mathcal{X}^\sharp[\, \mathtt{X} \mapsto [a, \min(b, d)], \; \mathtt{Y} \mapsto [\max(c, a), d] \,] & \text{otherwise} \end{cases}$$

$$C^\sharp[\![\, e \bowtie 0 \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \mathcal{X}^\sharp \quad \text{otherwise}$$

Note: fall-back operators

- $C^\sharp[\![\, e \bowtie 0 \,]\!]\, \mathcal{X}^\sharp = \mathcal{X}^\sharp$ is always sound.
- $C^\sharp[\![\, \mathtt{X} := e \,]\!]\, \mathcal{X}^\sharp = \mathcal{X}^\sharp[\mathtt{X} \mapsto \top^\sharp_b]$ is always sound.

Backward arithmetic and comparison operators

Given: sound backward arithmetic and comparison operators that refine their argument given a result.

i.e.

$$\mathcal{X}_b^{\sharp\prime} = \overleftarrow{\leq 0}_b^{\sharp}(\mathcal{X}_b^{\sharp}) \implies$$
$$\{\, x \in \gamma_b(\mathcal{X}_b^{\sharp}) \,|\, x \leq 0 \,\} \subseteq \gamma_b(\mathcal{X}_b^{\sharp\prime}) \subseteq \gamma_b(\mathcal{X}_b^{\sharp})$$

$$\mathcal{X}_b^{\sharp\prime} = \overleftarrow{-}_b^{\sharp}(\mathcal{X}_b^{\sharp}, \mathcal{R}_b^{\sharp}) \implies$$
$$\{\, x \,|\, x \in \gamma_b(\mathcal{X}_b^{\sharp}),\ -x \in \gamma_b(\mathcal{R}_b^{\sharp}) \,\} \subseteq \gamma_b(\mathcal{X}_b^{\sharp\prime}) \subseteq \gamma_b(\mathcal{X}_b^{\sharp})$$

$$(\mathcal{X}_b^{\sharp\prime}, \mathcal{Y}_b^{\sharp\prime}) = \overleftarrow{+}_b^{\sharp}(\mathcal{X}_b^{\sharp}, \mathcal{Y}_b^{\sharp}, \mathcal{R}_b^{\sharp}) \implies$$
$$\{\, x \in \gamma_b(\mathcal{X}_b^{\sharp}) \,|\, \exists y \in \gamma_b(\mathcal{Y}_b^{\sharp}),\ x + y \in \gamma_b(\mathcal{R}_b^{\sharp}) \,\} \subseteq \gamma_b(\mathcal{X}_b^{\sharp\prime}) \subseteq \gamma_b(\mathcal{X}_b^{\sharp})$$
$$\{\, y \in \gamma_b(\mathcal{Y}_b^{\sharp}) \,|\, \exists x \in \gamma_b(\mathcal{X}_b^{\sharp}),\ x + y \in \gamma_b(\mathcal{R}_b^{\sharp}) \,\} \subseteq \gamma_b(\mathcal{Y}_b^{\sharp\prime}) \subseteq \gamma_b(\mathcal{Y}_b^{\sharp})$$

$$\vdots$$

Note: best backward operators can be designed with α_b:

e.g. for $\overleftarrow{+}_b^{\sharp}$: $\mathcal{X}_b^{\sharp\prime} = \alpha_b(\{\, x \in \gamma_b(\mathcal{X}_b^{\sharp}) \,|\, \exists y \in \gamma_b(\mathcal{Y}_b^{\sharp}),\ x + y \in \gamma_b(\mathcal{R}_b^{\sharp}) \,\})$

Generic backward operator construction

Synthesizing (non optimal) backward arithmetic operators from forward arithmetic operators.

$$\overleftarrow{\leq 0}^\sharp_b(\mathcal{X}^\sharp_b) \stackrel{\text{def}}{=} \mathcal{X}^\sharp_b \cap^\sharp_b \,]-\infty,0]^\sharp_b$$

$$\overleftarrow{-}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{R}^\sharp_b) \stackrel{\text{def}}{=} \mathcal{X}^\sharp_b \cap^\sharp_b (-^\sharp_b\, \mathcal{R}^\sharp_b)$$

$$\overleftarrow{+}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b,\mathcal{R}^\sharp_b) \stackrel{\text{def}}{=} (\mathcal{X}^\sharp_b \cap^\sharp_b (\mathcal{R}^\sharp_b -^\sharp_b \mathcal{Y}^\sharp_b),\ \mathcal{Y}^\sharp_b \cap^\sharp_b (\mathcal{R}^\sharp_b -^\sharp_b \mathcal{X}^\sharp_b))$$

$$\overleftarrow{-}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b,\mathcal{R}^\sharp_b) \stackrel{\text{def}}{=} (\mathcal{X}^\sharp_b \cap^\sharp_b (\mathcal{R}^\sharp_b +^\sharp_b \mathcal{Y}^\sharp_b),\ \mathcal{Y}^\sharp_b \cap^\sharp_b (\mathcal{X}^\sharp_b -^\sharp_b \mathcal{R}^\sharp_b))$$

$$\overleftarrow{\times}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b,\mathcal{R}^\sharp_b) \stackrel{\text{def}}{=} (\mathcal{X}^\sharp_b \cap^\sharp_b (\mathcal{R}^\sharp_b \,/^\sharp_b\, \mathcal{Y}^\sharp_b),\ \mathcal{Y}^\sharp_b \cap^\sharp_b (\mathcal{R}^\sharp_b \,/^\sharp_b\, \mathcal{X}^\sharp_b))$$

$$\overleftarrow{/}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b,\mathcal{R}^\sharp_b) \stackrel{\text{def}}{=} (\mathcal{X}^\sharp_b \cap^\sharp_b (\mathcal{S}^\sharp_b \times^\sharp_b \mathcal{Y}^\sharp_b),\ \mathcal{Y}^\sharp_b \cap^\sharp_b ((\mathcal{X}^\sharp_b \,/^\sharp_b\, \mathcal{S}^\sharp_b) \cup^\sharp_b [0,0]^\sharp_b))$$

where $\mathcal{S}^\sharp_b = \begin{cases} \mathcal{R}^\sharp_b & \text{if } \mathbb{I} \neq \mathbb{Z} \\ \mathcal{R}^\sharp_b +^\sharp_b [-1,1]^\sharp_b & \text{if } \mathbb{I} = \mathbb{Z} \text{ (as / rounds)} \end{cases}$

Note: $\overleftarrow{\diamond}^\sharp_b(\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b,\mathcal{R}^\sharp_b) = (\mathcal{X}^\sharp_b,\mathcal{Y}^\sharp_b)$ is always sound (no refinement).

Interval backward operators

Applying the generic construction to the interval domain:

$$\overleftarrow{\leq 0}^\sharp_b([a,b]) \stackrel{\text{def}}{=} \begin{cases} [a, \min(b,0)] & \text{if } a \geq 0 \\ \bot^\sharp_b & \text{otherwise} \end{cases}$$

$$\overleftarrow{-}^\sharp_b([a,b],[r,s]) \stackrel{\text{def}}{=} [a,b] \cap^\sharp_b [-s,-r]$$

$$\overleftarrow{+}^\sharp_b([a,b],[c,d],[r,s]) \stackrel{\text{def}}{=} ([a,b] \cap^\sharp_b [r-d, s-c], \\ [c,d] \cap^\sharp_b [r-b, s-a])$$

...

Generic non-relational abstract test

Abstract test algorithm: $C^\sharp[\![\, e \bowtie 0 \,]\!]\, \mathcal{X}^\sharp$

Associate to each expression node an abstract value in $\mathcal{B}^\sharp$ using two traversals of the expression tree:

- first, a bottom-up evaluation using forward operators $\diamond_b^\sharp$,
- apply $\overleftarrow{\bowtie 0}_b^\sharp$ to the root,
- then, a top-down refinement using backward operators $\overleftarrow{\diamond}_b^\sharp$.

For each expression leaf, we get an abstract value $\mathcal{V}_b^\sharp$:

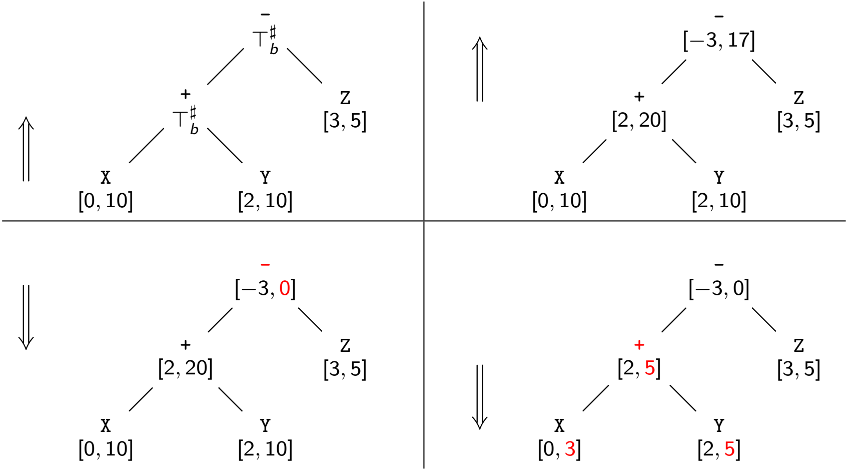
- for a variable V, replace $\mathcal{X}^\sharp(\mathtt{V})$ with $\mathcal{X}^\sharp(\mathtt{V}) \cap_b^\sharp \mathcal{V}_b^\sharp$,
- for a constant $[c, c']$, check that $[c, c']_b^\sharp \cap_b^\sharp \mathcal{V}_b^\sharp \neq \bot_b^\sharp$,
- $\Longrightarrow$ return $\bot^\sharp$ if some $\cap_b^\sharp \mathcal{V}_b^\sharp$ returns $\bot_b^\sharp$.

Improvement: local iterations [Gran92].

Interval test example

Example: $\mathsf{C}^\sharp[\![\, \mathtt{X} + \mathtt{Y} - \mathtt{Z} \le 0 \,]\!]\, \mathcal{X}^\sharp$

with $\mathcal{X}^\sharp = \{\, \mathtt{X} \mapsto [0,10], \mathtt{Y} \mapsto [2,10], \mathtt{Z} \mapsto [3,5] \,\}$



Generic non-relational backward assignment

Abstract function: $\overleftarrow{C}^\sharp[\![\, \mathtt{V} := e \,]\!]\,(\mathcal{X}^\sharp, \mathcal{R}^\sharp)$

over-approximates $\gamma(\mathcal{X}^\sharp) \cap \overleftarrow{C}[\![\, \mathtt{V} := e \,]\!]\,\gamma(\mathcal{R}^\sharp)$ given:

- an abstract pre-condition $\mathcal{X}^\sharp$ to refine,
- according to a given abstract post-condition $\mathcal{R}^\sharp$.

Algorithm: similar to the abstract test

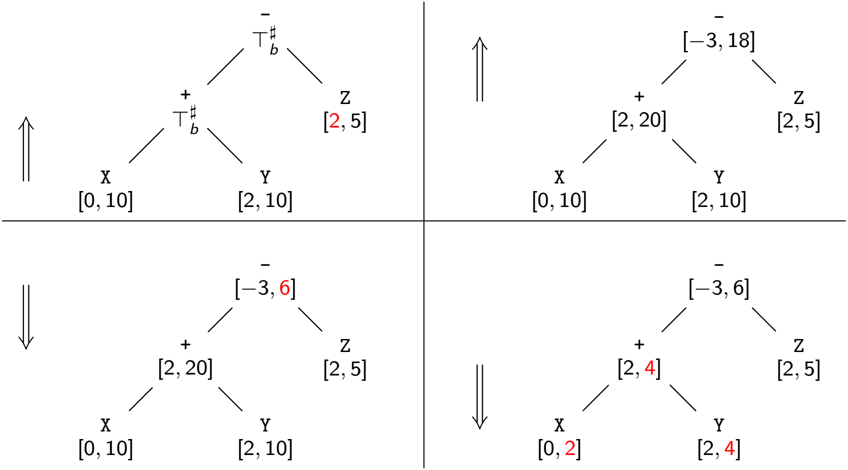
- annotate variable leaves based on $\mathcal{X}^\sharp \cap^\sharp (\mathcal{R}^\sharp[\mathtt{V} \mapsto \top^\sharp_b])$;
- evaluate bottom-up using forward operators $\diamond^\sharp_b$;
- intersect the root with $\mathcal{R}^\sharp(\mathtt{V})$;
- refine top-down using backward operators $\overleftarrow{\diamond}^\sharp_b$;
- return $\mathcal{X}^\sharp$ intersected with values at variable leaves.

Note:

- local iterations can also be used
- fallback: $\overleftarrow{C}^\sharp[\![\, \mathtt{V} := e \,]\!]\,(\mathcal{X}^\sharp, \mathcal{R}^\sharp) = \mathcal{X}^\sharp \cap^\sharp (\mathcal{R}^\sharp[\mathtt{V} \mapsto \top^\sharp_b])$

Interval backward assignment example

Example: $\overleftarrow{C}^\sharp[\![\, \texttt{X} := \texttt{X} + \texttt{Y} - \texttt{Z} \,]\!]\,(\mathcal{X}^\sharp, \mathcal{R}^\sharp)$
with $\mathcal{X}^\sharp = \{\, \texttt{X} \mapsto [0, 10], \texttt{Y} \mapsto [2, 10], \texttt{Z} \mapsto [1, 5] \,\}$
and $\mathcal{R}^\sharp = \{\, \texttt{X} \mapsto [-6, 6], \texttt{Y} \mapsto [2, 10], \texttt{Z} \mapsto [2, 6] \,\}$



Interval widening

Widening on non-relational domains:

Given a value widening $\triangledown_b\colon \mathcal{B}^\sharp \times \mathcal{B}^\sharp \to \mathcal{B}^\sharp$,
we extend it point-wisely into a widening $\triangledown\colon \mathcal{D}^\sharp \times \mathcal{D}^\sharp \to \mathcal{D}^\sharp$:

$$\mathcal{X}^\sharp \triangledown \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \lambda \mathtt{V}.(\mathcal{X}^\sharp(\mathtt{V}) \triangledown_b \mathcal{Y}^\sharp(\mathtt{V}))$$

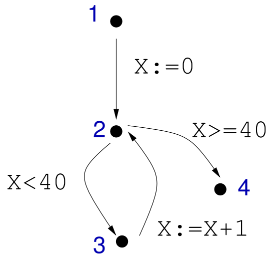
Interval widening example:

$$\begin{array}{lllll}
\bot^\sharp & \triangledown_b & X^\sharp & \stackrel{\text{def}}{=} & X^\sharp \\
[a,b] & \triangledown_b & [c,d] & \stackrel{\text{def}}{=} & \left[\begin{cases} a & \text{if } a \leq c \\ -\infty & \text{otherwise} \end{cases}, \begin{cases} b & \text{if } b \geq d \\ +\infty & \text{otherwise} \end{cases} \right]
\end{array}$$

Unstable bounds are set to $\pm\infty$.

Analysis with widening example

Analysis example with $\mathcal{W} = \{2\}$



ℓ	$\mathcal{X}_\ell^{\sharp 0}$	$\mathcal{X}_\ell^{\sharp 1}$	$\mathcal{X}_\ell^{\sharp 2}$	$\mathcal{X}_\ell^{\sharp 3}$	$\mathcal{X}_\ell^{\sharp 4}$	$\mathcal{X}_\ell^{\sharp 5}$
1	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$
2 $\triangledown$	$\bot^\sharp$	$= 0$	$= 0$	≥ 0	≥ 0	≥ 0
3	$\bot^\sharp$	$\bot^\sharp$	$= 0$	$= 0$	$\in [0, 39]$	$\in [0, 39]$
4	$\bot^\sharp$	$\bot^\sharp$	$\bot^\sharp$	$\bot^\sharp$	≥ 40	≥ 40

More precisely, at the widening point:

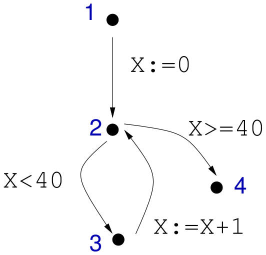
$$
\begin{array}{lclclcl}
\mathcal{X}_2^{\sharp 1} & = & \bot^\sharp \;\triangledown_b\, ([0,0] \cup_b^\sharp \bot^\sharp) & = & \bot^\sharp \;\triangledown_b\, [0,0] & = & [0,0] \\
\mathcal{X}_2^{\sharp 2} & = & [0,0] \;\triangledown_b\, ([0,0] \cup_b^\sharp \bot^\sharp) & = & [0,0] \;\triangledown_b\, [0,0] & = & [0,0] \\
\mathcal{X}_2^{\sharp 3} & = & [0,0] \;\triangledown_b\, ([0,0] \cup_b^\sharp [1,1]) & = & [0,0] \;\triangledown_b\, [0,1] & = & [0,+\infty[\\
\mathcal{X}_2^{\sharp 4} & = & [0,+\infty[\;\triangledown_b\, ([0,0] \cup_b^\sharp [1,40]) & = & [0,+\infty[\;\triangledown_b\, [0,40] & = & [0,+\infty[
\end{array}
$$

Note that the most precise interval abstraction would be $\texttt{X} \in [0, 40]$ at 2, and $\texttt{X} = 40$ at 4.

Influence of the widening point and iteration strategy

Changing $\mathcal{W}$ changes the analysis result

Example: The analysis is less precise for $\mathcal{W} = \{3\}$.



ℓ	$\mathcal{X}_\ell^{\sharp 1}$	$\mathcal{X}_\ell^{\sharp 2}$	$\mathcal{X}_\ell^{\sharp 3}$	$\mathcal{X}_\ell^{\sharp 4}$	$\mathcal{X}_\ell^{\sharp 5}$	$\mathcal{X}_\ell^{\sharp 6}$
1	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$
2	$= 0$	$= 0$	$\in [0,1]$	$\in [0,1]$	≥ 0	≥ 0
3 $\triangledown$	$\bot^\sharp$	$= 0$	$= 0$	≥ 0	≥ 0	≥ 0
4	$\bot^\sharp$	$\bot^\sharp$	$\bot^\sharp$	$\bot^\sharp$	$\bot^\sharp$	≥ 40

Intuition: extrapolation to $+\infty$ is no longer contained by the tests.

Chaotic iterations

Changing the iteration order changes the analysis result in the presence of a widening.

Narrowing

Using a widening makes the analysis less precise.

Some precision can be retrieved by using a narrowing $\triangle$.

Definition: narrowing $\triangle$

Binary operator $\mathcal{D}^\sharp \times \mathcal{D}^\sharp \to \mathcal{D}^\sharp$ such that:

- $(\mathcal{X}^\sharp \cap^\sharp \mathcal{Y}^\sharp) \sqsubseteq (\mathcal{X}^\sharp \triangle \mathcal{Y}^\sharp) \sqsubseteq \mathcal{X}^\sharp$,
- for all sequences $(\mathcal{X}_i^\sharp)$, the decreasing sequence $(\mathcal{Y}_i^\sharp)$ defined by $\begin{cases} \mathcal{Y}_0^\sharp & \stackrel{\text{def}}{=} \mathcal{X}_0^\sharp \\ \mathcal{Y}_{i+1}^\sharp & \stackrel{\text{def}}{=} \mathcal{Y}_i^\sharp \triangle \mathcal{X}_{i+1}^\sharp \end{cases}$ is stationary.

This is not the dual of a widening!

Narrowing examples

Trivial narrowing:

$\mathcal{X}^\sharp \mathbin{\triangle} \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \mathcal{X}^\sharp$ is a correct narrowing.

Finite-time intersection narrowing:

$$\mathcal{X}^{\sharp i} \mathbin{\triangle} \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \begin{cases} \mathcal{X}^{\sharp i} \cap^\sharp \mathcal{Y}^\sharp & \text{if } i \leq N \\ \mathcal{X}^{\sharp i} & \text{if } i > N \end{cases}$$

Interval narrowing:

$$[a,b] \mathbin{\triangle_b} [c,d] \stackrel{\text{def}}{=} \left[\begin{cases} c & \text{if } a = -\infty \\ a & \text{otherwise} \end{cases}, \begin{cases} d & \text{if } b = +\infty \\ b & \text{otherwise} \end{cases} \right]$$

(refine only infinite bounds)

Point-wise extension to $\mathcal{D}^\sharp$: $\mathcal{X}^\sharp \mathbin{\triangle} \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \lambda \mathtt{V}.(\mathcal{X}^\sharp(\mathtt{V}) \mathbin{\triangle_b} \mathcal{Y}^\sharp(\mathtt{V}))$

Iterations with narrowing

Let $\mathcal{X}_\ell^{\sharp\delta}$ be the result after widening stabilisation, *i.e.*:

$$\mathcal{X}_\ell^{\sharp\delta} \sqsupseteq \begin{cases} \top^\sharp & \text{if } \ell = e \\ \bigcup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{X}_{\ell'}^{\sharp\delta} & \text{if } \ell \neq e \end{cases}$$

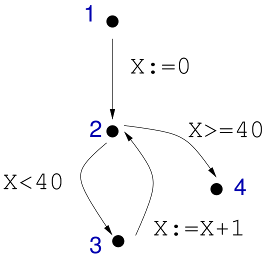
The following sequence is computed:

$$\mathcal{Y}_\ell^{\sharp 0} \stackrel{\text{def}}{=} \mathcal{X}_\ell^{\sharp\delta} \qquad \mathcal{Y}_\ell^{\sharp i+1} \stackrel{\text{def}}{=} \begin{cases} \top^\sharp & \text{if } \ell = e \\ \bigcup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{Y}_{\ell'}^{\sharp i} & \text{if } \ell \notin \mathcal{W} \\ \mathcal{Y}_\ell^{\sharp i} \,\triangle \bigcup^\sharp_{(\ell',c,\ell)\in A} \mathsf{C}^\sharp[\![\, c \,]\!]\, \mathcal{Y}_{\ell'}^{\sharp i} & \text{if } \ell \in \mathcal{W} \end{cases}$$

- the sequence $(\mathcal{Y}_\ell^{\sharp i})$ is decreasing and converges in finite time,
- all $(\mathcal{Y}_\ell^{\sharp i})$ are solutions of the abstract semantic system.

Analysis with narrowing example

Example with $\mathcal{W} = \{2\}$



ℓ	$\mathcal{Y}_\ell^{\sharp 0}$	$\mathcal{Y}_\ell^{\sharp 1}$	$\mathcal{Y}_\ell^{\sharp 2}$	$\mathcal{Y}_\ell^{\sharp 3}$
1	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$
2 $\triangle$	≥ 0	$\in [0, 40]$	$\in [0, 40]$	$\in [0, 40]$
3	$\in [0, 39]$	$\in [0, 39]$	$\in [0, 39]$	$\in [0, 39]$
4	≥ 40	≥ 40	$= 40$	$= 40$

Narrowing at 2 gives:

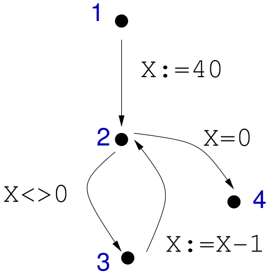
$$\mathcal{Y}_2^{\sharp 1} = [0, +\infty[\;\triangle_b\; ([0,0] \cup_b^\sharp [1,40]) = [0, +\infty[\;\triangle_b\; [0,40] = [0,40]$$
$$\mathcal{Y}_2^{\sharp 2} = [0, 40] \;\triangle_b\; ([0,0] \cup_b^\sharp [1,40]) = [0, 40] \;\triangle_b\; [0,40] = [0,40]$$

Then $\mathcal{Y}_2^{\sharp 2} : \mathtt{X} \in [0, 40]$ gives $\mathcal{Y}_4^{\sharp 3} : \mathtt{X} = 40$.

We found the most precise invariants!

Improving the widening

Example of imprecise analysis



ℓ	intervals with $\triangledown_b$	extended signs	intervals with $\triangledown'_b$
1	$\top^\sharp$	$\top^\sharp$	$\top^\sharp$
2 $\triangledown$	$\mathtt{X} \leq 40$	$\mathtt{X} \geq 0$	$\mathtt{X} \in [0, 40]$
3	$\mathtt{X} \leq 40$	$\mathtt{X} > 0$	$\mathtt{X} \in [0, 40]$
4	$\mathtt{X} = 0$	$\mathtt{X} = 0$	$\mathtt{X} = 0$

The interval domain cannot prove that $\mathtt{X} \geq 0$ at 2,
while the (less powerful) sign domain can!

Solution: improve the interval widening

$$[a, b] \; \triangledown'_b \; [c, d] \stackrel{\text{def}}{=} \left[\begin{cases} a & \text{if } a \leq c \\ 0 & \text{if } 0 \leq c < a \\ -\infty & \text{otherwise} \end{cases}, \begin{cases} b & \text{if } b \geq d \\ 0 & \text{if } 0 \geq b > d \\ +\infty & \text{otherwise} \end{cases} \right]$$

($\triangledown'_b$ checks the stability of 0)

Widening with thresholds

Analysis problem:

```
X:=0;
while • 1=1 do
  if [0,1]=0 then
    X:=X+1;
    if X>40 then X:=0 fi
  fi
done
```

We wish to prove that $X \in [0, 40]$ at •.

- Widening at • finds the loop invariant $\mathtt{X} \in [0, +\infty[$.
  $\mathcal{X}^\sharp_\bullet = [0,0] \triangledown_b ([0,0] \cup^\sharp [0,1]) = [0,0] \triangledown_b [0,1] = [0,+\infty[$
- Narrowing is unable to refine the invariant:
  $\mathcal{Y}^\sharp_\bullet = [0,+\infty[\triangle_b ([0,0] \cup^\sharp [0,+\infty[) = [0,+\infty[$

  (the code that limits X is not executed at every loop iteration)

Widening with thresholds (cont.)

Solution:

Choose a finite set T of thresholds containing $+\infty$ and $-\infty$.

Definition: widening with thresholds $\triangledown_b^T$

$$[a,b] \triangledown_b^T [c,d] \stackrel{\text{def}}{=} \left[\begin{cases} a & \text{if } a \le c \\ \max \{x \in T \mid x \le c\} & \text{otherwise} \end{cases}, \begin{cases} b & \text{if } b \ge d \\ \min \{x \in T \mid x \ge d\} & \text{otherwise} \end{cases} \right]$$

The widening tests and stops at the first stable bound in T.

Widening with thresholds (cont.)

Applications:

- On the previous example, we find:
  $\texttt{X} \in [\,0,\ \min\{x \in T \mid x \geq 40\}\,]$.
- Useful when it is easy to find a 'good' set T.
  Example: array bound-checking
- Useful if an over-approximation of the bound is sufficient.
  Example: arithmetic overflow checking

Limitations: only works if some non-∞ bound in T is stable.

Example: with $T = \{\,5, 15\,\}$

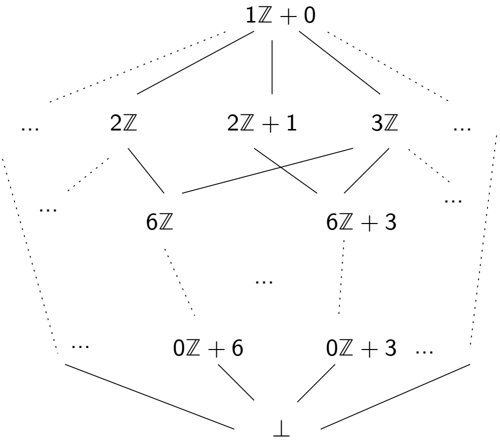
`while 1=1 do`
`  X:=X+1;`
`  if X>10 then X=0 fi`
`done`
	`while 1=1 do`
	`  X:=X+1;`
	`  if X<>10 then X=0 fi`
	`done`

15 is stable	no stable bound

The congruence domain

The congruence lattice

$$\mathcal{B}^\sharp \stackrel{\text{def}}{=} \{\, (a\mathbb{Z} + b) \mid a \in \mathbb{N},\ b \in \mathbb{Z} \,\} \cup \{\, \perp_b^\sharp \,\}$$



Introduced by Granger [Gran89].

We take $\mathbb{I} = \mathbb{Z}$.

The congruence lattice (cont.)

Concretization:

$$\gamma_b(\mathcal{X}_b^\sharp) \stackrel{\text{def}}{=} \begin{cases} \{\, ak+b \mid k \in \mathbb{Z} \,\} & \text{if } \mathcal{X}_b^\sharp = (a\mathbb{Z}+b) \\ \emptyset & \text{if } \mathcal{X}_b^\sharp = \bot_b^\sharp \end{cases}$$

Note that $\gamma(0\mathbb{Z}+b) = \{b\}$.
γ_b is not injective: $\gamma_b(2\mathbb{Z}+1) = \gamma_b(2\mathbb{Z}+3)$.

Definitions:

Given $x, x' \in \mathbb{Z}$, $y, y' \in \mathbb{N}$, we define:

- $y/y' \stackrel{\text{def}}{\iff} y$ divides y' $(\exists k \in \mathbb{N},\ y' = ky)$ (note that $\forall y$: $y/0$)
- $x \equiv x'\ [y] \stackrel{\text{def}}{\iff} y / |x - x'|$ (in particular, $x \equiv x'\ [0] \iff x = x'$)
- $\vee$ is the LCM, extended with $y \vee 0 \stackrel{\text{def}}{=} 0 \vee y \stackrel{\text{def}}{=} 0$
- $\wedge$ is the GCD, extended with $y \wedge 0 \stackrel{\text{def}}{=} 0 \wedge y \stackrel{\text{def}}{=} y$

$(\mathbb{N}, /, \vee, \wedge, 1, 0)$ is a complete distributive lattice.

Abstract congruence operators

Complete lattice structure on $\mathcal{B}^\sharp$:

- $(a\mathbb{Z}+b) \sqsubseteq_b (a'\mathbb{Z}+b') \stackrel{\text{def}}{\iff} a'/a$ and $b \equiv b'\ [a']$
- $\top_b^\sharp \stackrel{\text{def}}{=} (1\mathbb{Z}+0)$
- $(a\mathbb{Z}+b) \cup_b^\sharp (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} (a \wedge a' \wedge |b-b'|)\mathbb{Z}+b$
- $(a\mathbb{Z}+b) \cap_b^\sharp (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} \begin{cases} (a \vee a')\mathbb{Z}+b'' & \text{if } b \equiv b'\ [a \wedge a'] \\ \perp_b^\sharp & \text{otherwise} \end{cases}$

  b'' such that $b'' \equiv b\ [a \vee a'] \equiv b'\ [a \vee a']$ is given by Bezout's Theorem.

Galois connection:

$$\alpha_b(\mathcal{X}) = \bigcup_{c\in\mathcal{X}}{}_b^\sharp\, (0\mathbb{Z}+c)$$

(up to equivalence $a\mathbb{Z}+b \equiv a'\mathbb{Z}+b' \stackrel{\text{def}}{\iff} a = a' \wedge b \equiv b'\ [a]$)

Abstract congruence operators (cont.)

Arithmetic operators:

$$[c, c']^\sharp_b \stackrel{\text{def}}{=} \begin{cases} 0\mathbb{Z} + c & \text{if } c = c' \\ \top^\sharp_b & \text{otherwise} \end{cases}$$

$$-^\sharp_b\, (a\mathbb{Z} + b) \stackrel{\text{def}}{=} a\mathbb{Z} + (-b)$$

$$(a\mathbb{Z} + b) +^\sharp_b (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} (a \wedge a')\mathbb{Z} + (b + b')$$

$$(a\mathbb{Z} + b) -^\sharp_b (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} (a \wedge a')\mathbb{Z} + (b - b')$$

$$(a\mathbb{Z} + b) \times^\sharp_b (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} (aa' \wedge ab' \wedge a'b)\mathbb{Z} + bb'$$

$$(a\mathbb{Z} + b) \;/^\sharp_b\; (a'\mathbb{Z} + b') \stackrel{\text{def}}{=}$$

$$\begin{cases} \bot^\sharp_b & \text{if } a'\mathbb{Z} + b' = 0\mathbb{Z} + 0 \\ (a/|b'|)\mathbb{Z} + (b/b') & \text{if } a' = 0,\ b' \neq 0,\ b'|a,\ \text{and } b'|b \\ \top^\sharp_b & \text{otherwise (not optimal)} \end{cases}$$

Abstract congruence operators (cont.)

Test operators:

$$\overleftarrow{\leq 0}^\sharp_b\,(a\mathbb{Z}+b) \quad \stackrel{\text{def}}{=} \quad \begin{cases} \perp^\sharp_b & \text{if } a=0,\ b>0 \\ a\mathbb{Z}+b & \text{otherwise} \end{cases}$$

$$\vdots$$

Note: better than the generic $\overleftarrow{\leq 0}^\sharp_b\,(\mathcal{X}^\sharp_b) \stackrel{\text{def}}{=} \mathcal{X}^\sharp_b \cap^\sharp_b\,]-\infty,0]^\sharp_b = \mathcal{X}^\sharp_b$

Extrapolation operators:

- no infinite increasing chain $\Longrightarrow$ no need for $\triangledown$
- infinite decreasing chains $\Longrightarrow$ $\triangle$ needed

$$(a\mathbb{Z}+b)\ \triangle_b\ (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} \begin{cases} a'\mathbb{Z}+b' & \text{if } a=1 \\ a\mathbb{Z}+b & \text{otherwise} \end{cases}$$

Note: $\mathcal{X}^\sharp \triangle \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \mathcal{X}^\sharp$ is always a narrowing.

Congruence analysis example

```
X:=0; Y:=2;
while • X<40 do
  X:=X+2;
  if X<5 then Y:=Y+18 fi;
  if X>8 then Y:=Y-30 fi
done
```

We find, at •, the loop invariant $\begin{cases} \mathtt{X} \in 2\mathbb{Z} \\ \mathtt{Y} \in 6\mathbb{Z} + 2 \end{cases}$

Reduced products of domains

Non-reduced product of domains

Product representation:

Cartesian product $\mathcal{D}^\sharp_{1\times 2}$ of $\mathcal{D}^\sharp_1$ and $\mathcal{D}^\sharp_2$:

- $\mathcal{D}^\sharp_{1\times 2} \stackrel{\text{def}}{=} \mathcal{D}^\sharp_1 \times \mathcal{D}^\sharp_2$
- $\gamma_{1\times 2}(\mathcal{X}^\sharp_1, \mathcal{X}^\sharp_2) \stackrel{\text{def}}{=} \gamma_1(\mathcal{X}^\sharp_1) \cap \gamma_2(\mathcal{X}^\sharp_2)$
- $\alpha_{1\times 2}(\mathcal{X}) \stackrel{\text{def}}{=} (\alpha_1(\mathcal{X}), \alpha_2(\mathcal{X}))$
- $(\mathcal{X}^\sharp_1, \mathcal{X}^\sharp_2) \sqsubseteq_{1\times 2} (\mathcal{Y}^\sharp_1, \mathcal{Y}^\sharp_2) \stackrel{\text{def}}{\iff} X^\sharp_1 \sqsubseteq_1 \mathcal{Y}^\sharp_1$ and $X^\sharp_2 \sqsubseteq_2 \mathcal{Y}^\sharp_2$

Abstract operators: performed in parallel on both components:

- $(\mathcal{X}^\sharp_1, \mathcal{X}^\sharp_2) \cup^\sharp_{1\times 2} (\mathcal{Y}^\sharp_1, \mathcal{Y}^\sharp_2) \stackrel{\text{def}}{=} (\mathcal{X}^\sharp_1 \cup^\sharp_1 \mathcal{Y}^\sharp_1, \mathcal{X}^\sharp_2 \cup^\sharp_2 \mathcal{Y}^\sharp_2)$
  and the same for $\triangledown^\sharp_{1\times 2}$ and $\triangle^\sharp_{1\times 2}$
- $C^\sharp[\![\, c \,]\!]_{1\times 2}(\mathcal{X}^\sharp_1, \mathcal{X}^\sharp_2) \stackrel{\text{def}}{=} (C^\sharp[\![\, c \,]\!]_1(\mathcal{X}^\sharp_1), C^\sharp[\![\, c \,]\!]_2(\mathcal{X}^\sharp_2))$

Non-reduced product example

The product analysis is no more precise than two separate analyses.

Example: interval–congruence product:

```
 X:=1;
 while X-10<=0 do
   X:=X+2
 done;
•if X-12>=0 then♦ X:=0★ fi
```

	interval	congruence	product γ
•	$\mathtt{X} \in [11, 12]$	$\mathtt{X} \equiv 1\ [2]$	$\mathtt{X} = 11$
♦	$\mathtt{X} = 12$	$\mathtt{X} \equiv 1\ [2]$	$\emptyset$
★	$\mathtt{X} = 0$	$\mathtt{X} = 0$	$\mathtt{X} = 0$

We cannot prove that the `if` branch is never taken!

Fully-reduced product

Definition:

Given the Galois connections (α_1, γ_1) and (α_2, γ_2) on $\mathcal{D}_1^\sharp$ and $\mathcal{D}_2^\sharp$ we define the reduction operator ρ as:

$$\rho : \mathcal{D}_{1\times 2}^\sharp \to \mathcal{D}_{1\times 2}^\sharp$$
$$\rho(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \stackrel{\text{def}}{=} (\alpha_1(\gamma_1(\mathcal{X}_1^\sharp) \cap \gamma_2(\mathcal{X}_2^\sharp)), \alpha_2(\gamma_1(\mathcal{X}_1^\sharp) \cap \gamma_2(\mathcal{X}_2^\sharp)))$$

ρ propagates information between domains.

Application:

We can reduce the result of each abstract operator, except $\triangledown$:

- $(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \cup_{1\times 2}^\sharp (\mathcal{Y}_1^\sharp, \mathcal{Y}_2^\sharp) \stackrel{\text{def}}{=} \rho(\mathcal{X}_1^\sharp \cup_1^\sharp \mathcal{Y}_1^\sharp, \mathcal{X}_2^\sharp \cup_2^\sharp \mathcal{Y}_2^\sharp)$,
- $C^\sharp[\![c]\!]_{1\times 2}(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \stackrel{\text{def}}{=} \rho(C^\sharp[\![c]\!]_1(\mathcal{X}_1^\sharp), C^\sharp[\![c]\!]_2(\mathcal{X}_2^\sharp))$.

We refrain from reducing after a widening $\triangledown$, this may jeopardize the convergence (octagon domain example).

Fully-reduced product example

Reduction example: between the interval and congruence domains:

Noting:
$$a' \stackrel{\text{def}}{=} \min \{ x \geq a \mid x \equiv d\ [c] \}$$
$$b' \stackrel{\text{def}}{=} \max \{ x \leq b \mid x \equiv d\ [c] \}$$

We get:

$$\rho_b([a,b], c\mathbb{Z}+d) \stackrel{\text{def}}{=} \begin{cases} (\perp_b^\sharp, \perp_b^\sharp) & \text{if } a' > b' \\ ([a',a'], 0\mathbb{Z}+a') & \text{if } a' = b' \\ ([a',b'], c\mathbb{Z}+d) & \text{if } a' < b' \end{cases}$$

extended point-wisely to ρ on $\mathcal{D}^\sharp$.

Application:

- $\rho_b([10,11],\ 2\mathbb{Z}+1) = ([11,11],\ 0\mathbb{Z}+11)$
  (proves that the branch is never taken on our example)
- $\rho_b([1,3],\ 4\mathbb{Z}) = (\perp_b^\sharp,\ \perp_b^\sharp)$

Partially-reduced product

Definition: of a partial reduction:

any function $\rho : \mathcal{D}^\sharp_{1\times 2} \to \mathcal{D}^\sharp_{1\times 2}$ such that:

$$(\mathcal{Y}_1^\sharp, \mathcal{Y}_2^\sharp) = \rho(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \implies \begin{cases} \gamma_{1\times 2}(\mathcal{Y}_1^\sharp, \mathcal{Y}_2^\sharp) = \gamma_{1\times 2}(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \\ \gamma_1(\mathcal{Y}_1^\sharp) \subseteq \gamma_1(\mathcal{X}_1^\sharp) \\ \gamma_2(\mathcal{Y}_2^\sharp) \subseteq \gamma_2(\mathcal{X}_2^\sharp) \end{cases}$$

Useful when:

- there is no Galois connection, or
- a full reduction exists but is expensive to compute.

Partial reduction example:

$$\rho(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \stackrel{\text{def}}{=} \begin{cases} (\perp^\sharp, \perp^\sharp) & \text{if } \mathcal{X}_1^\sharp = \perp^\sharp \text{ or } \mathcal{X}_2^\sharp = \perp^\sharp \\ (\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) & \text{otherwise} \end{cases}$$

(works on all domains)

For more complex examples, see [Blan03].

Bibliography

Bibliography

[Anco10] **C. Ancourt, F. Coelho & F. Irigoin**. *A modular static analysis approach to affine loop invariants detection.* In Proc. NSAD'10, ENTCS, Elsevier, 2010.

[Berd07] **J. Berdine, A. Chawdhary, B. Cook, D. Distefano & P. O'Hearn**. *Variance analyses from invariances analyses.* In Proc. POPL'07 211–224, ACM, 2007.

[Blan03] **B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux & X. Rival**. *A static analyzer for large safety-critical software.* In Proc. PLDI'03, 196–207, ACM, 2003.

[Bour93a] **F. Bourdoncle**. *Efficient chaotic iteration strategies with widenings.* In Proc. FMPA'93, LNCS 735, 128–141, Springer, 1993.

[Bour93b] **F. Bourdoncle**. *Assertion-based debugging of imperative programs by abstract interpretation.* In Proc. ESEC'93, 501–516, Springer, 1993.

Bibliography (cont.)

[Cous76] **P. Cousot & R. Cousot**. *Static determination of dynamic properties of programs.* In Proc. ISP'76, Dunod, 1976.

[Dor01] **N. Dor, M. Rodeh & M. Sagiv**. *Cleanness checking of string manipulations in C programs via integer analysis.* In Proc. SAS'01, LNCS 2126, 194–212, Springer, 2001.

[Girb06] **S. Girbal, N. Vasilache, C. Bastoul, A. Cohen, D. Parello, M. Sigler & O. Temam**. *Semi-automatic composition of loop transformations for deep parallelism and memory hierarchies.* In J. of Parallel Prog., 34(3):261–317, 2006.

[Gran89] **P. Granger**. *Static analysis of arithmetical congruences.* In JCM, 3(4–5):165–190, 1989.

[Gran92] **P. Granger**. *Improving the results of static analyses of programs by local decreasing iterations.* In Proc. FSTTCSC'92, LNCS 652, 68–79, Springer, 1992.

Bibliography (cont.)

[Gran97] **P. Granger**. *Static analyses of congruence properties on rational numbers.* In Proc. SAS'97, LNCS 1302, 278–292, Springer, 1997.

[Jean09] **B. Jeannet & A. Miné**. *Apron: A library of numerical abstract domains for static analysis.* In Proc. CAV'09, LNCS 5643, 661–667, Springer, 2009, `http://apron.cri.ensmp.fr/library`.

[Mine06] **A. Miné**. *Field-sensitive value analysis of embedded C programs with union types and pointer arithmetics.* In Proc. LCTES'06, 54–63, ACM, 2006.

[Vene02] **A. Venet**. *Nonuniform alias analysis of recursive data structures and arrays.* In Proc. SAS'02, LNCS 2477, 36–51, Springer, 2002.