

Non-Relational Numerical Abstract Domains

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

Antoine Miné

Year 2019–2020

Course 03

25 September 2019

Outline

- Concrete semantics (reminders)
- Abstract domains and abstract solving
- Non-relational numerical abstract domains
 - generic Cartesian abstraction
 - sign domain
 - constant domain
 - interval domain
 - congruence domain
- Reduced product of domains
- Bibliography

Next week: relational abstract domains

Concrete semantics (reminders)

Syntax of a toy-language

Simple numeric programs:

- fixed, **finite** set of variables \mathbb{V}
- with value in some **numeric set** $\mathbb{I} \stackrel{\text{def}}{=} \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$
- programs as **CFG**: (\mathcal{L}, e, x, A)
with nodes \mathcal{L} , entry $e \in \mathcal{L}$, exit $x \in \mathcal{L}$, and arcs $A \subseteq \mathcal{L} \times \text{com} \times \mathcal{L}$

Atomic commands:

$\text{com} ::=$	$V \leftarrow \text{exp}$	assignment into $V \in \mathbb{V}$
	$\text{exp} \bowtie 0$	test, $\bowtie \in \{=, <, >, \leq, \geq, \neq\}$

Arithmetic expressions:

$\text{exp} ::=$	V	variable $V \in \mathbb{V}$
	$-\text{exp}$	negation
	$\text{exp} \diamond \text{exp}$	binary operation: $\diamond \in \{+, -, \times, /\}$
	$[c, c']$	constant range, $c, c' \in \mathbb{I} \cup \{\pm\infty\}$
	c	constant, shorthand for $[c, c]$

Expression semantics

Expression semantics: $E[e]: \mathcal{E} \rightarrow \mathcal{P}(\mathbb{I})$

where $\mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \rightarrow \mathbb{I}$.

The evaluation of e in $\rho \in \mathcal{E}$ gives a **set** of values:

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

Forward semantics: state reachability

Transfer functions: $C[\text{com}]: \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{E})$

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

Fixpoint semantics: $(\mathcal{X}_\ell)_{\ell \in \mathcal{L}}: \mathcal{P}(\mathcal{E})$

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

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

$\mathcal{D} \stackrel{\text{def}}{=} (\mathcal{P}(\mathcal{E}), \subseteq, \cup, \cap, \emptyset, \mathcal{E})$ is a complete lattice,

each $M_\ell: \mathcal{X}_\ell \mapsto \bigcup_{(\ell', c, \ell) \in A} C[c] \mathcal{X}_{\ell'}$ is monotonic in \mathcal{D} .

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

Resolution

Resolution by increasing iterations:

$$\left\{ \begin{array}{l} \mathcal{X}_e^0 \stackrel{\text{def}}{=} \mathcal{E} \\ \mathcal{X}_{\ell \neq e}^0 \stackrel{\text{def}}{=} \emptyset \end{array} \right. \quad \left\{ \begin{array}{l} \mathcal{X}_e^{n+1} \stackrel{\text{def}}{=} \mathcal{E} \\ \mathcal{X}_{\ell \neq e}^{n+1} \stackrel{\text{def}}{=} \bigcup_{(\ell', c, \ell) \in A} C[[c]] \mathcal{X}_{\ell'}^n \end{array} \right.$$

Kleene theorem:

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

Backward refinement: state co-reachability

Semantics of commands: $\overleftarrow{C}[[c]]: \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{E})$

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

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

Refinement: given:

- a solution $(\mathcal{X}_\ell)_{\ell \in \mathcal{L}}$ of the forward system
- an output **criterion** \mathcal{Y} at exit node x

compute a least fixpoint by **decreasing iterations** [Bour93b]

$$\begin{cases} \mathcal{Y}_x^0 & \stackrel{\text{def}}{=} & \mathcal{X}_x \cap \mathcal{Y} \\ \mathcal{Y}_{\ell \neq x}^0 & \stackrel{\text{def}}{=} & \mathcal{X}_\ell \end{cases}$$

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

Limit to automation

We wish to perform **automatic** numerical invariant discovery.

Theoretical problems

- the elements of $\mathcal{P}(\mathbb{V} \rightarrow \mathbb{I})$ are **not computer representable**
- the transfer functions $C[[c]]$, $\overleftarrow{C}[[c]]$ are **not computable**
- the lattice iterations in $\mathcal{P}(\mathcal{E})$ 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 $\mathcal{P}(\mathbb{V} \rightarrow \mathbb{I})$ in extension is expensive
- computing $C[[c]]$, $\overleftarrow{C}[[c]]$ explicitly is expensive
- the lattice $\mathcal{P}(\mathbb{V} \rightarrow \mathbb{I})$ has a large height (\Rightarrow many iterations)

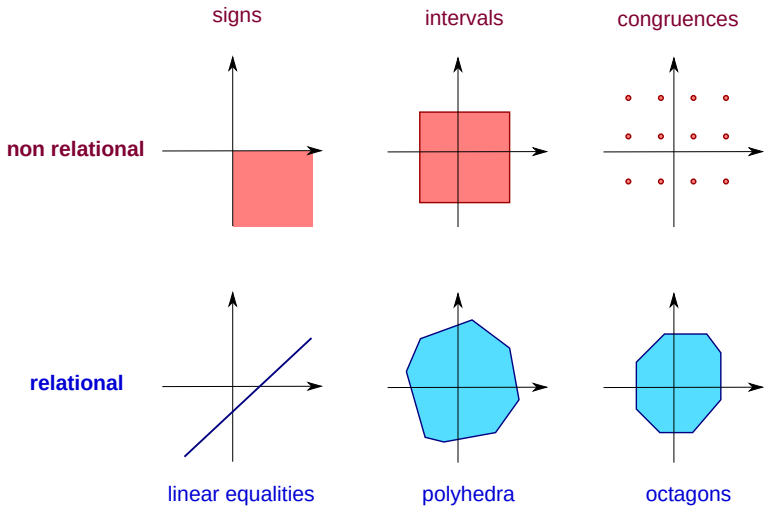
Abstraction

Numerical abstract domains

A **numerical abstract domain** is given by:

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

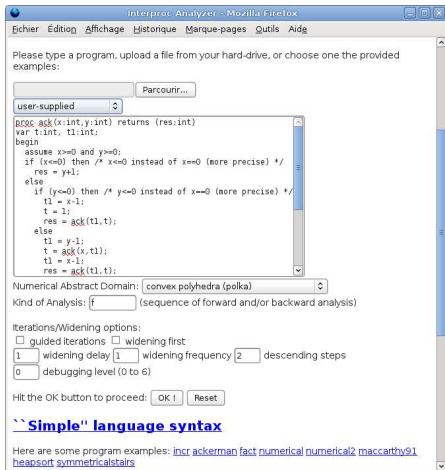
Numerical abstract domain examples



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>

Numerical abstract domains (cont.)

Representation: given by

- a set $\mathcal{D}^\#$ of machine-representable **abstract environments**,
- a **partial order** $(\mathcal{D}^\#, \sqsubseteq, \perp^\#, \top^\#)$
relating the amount of information given by abstract elements,
- a **concretization** function $\gamma: \mathcal{D}^\# \rightarrow \mathcal{P}(\mathcal{E})$
giving a concrete meaning to each abstract element,
- an abstraction function α forming a Galois connection (α, γ) is optional.

Required algebraic properties:

- γ should be **monotonic**: $\mathcal{X}^\# \sqsubseteq \mathcal{Y}^\# \implies \gamma(\mathcal{X}^\#) \subseteq \gamma(\mathcal{Y}^\#)$,
- $\gamma(\perp^\#) = \emptyset$,
- $\gamma(\top^\#) = \mathcal{E}$.

Note: γ need not be one-to-one.

Numerical abstract domains (cont.)

Abstract operators: we require:

- sound, effective, abstract **transfer functions** $C^\# \llbracket c \rrbracket$, $\overleftarrow{C}^\# \llbracket c \rrbracket$ for all commands c ,
- sound, effective, abstract **set operators** $\cup^\#$, $\cap^\#$,
- an algorithm to decide the **ordering** \sqsubseteq .

Soundness criterion:

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

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

$F^\#(\mathcal{X}_1^\#, \dots, \mathcal{X}_n^\#) = \alpha(F(\gamma(\mathcal{X}_1^\#), \dots, \gamma(\mathcal{X}_n^\#)))$ is optional.

Both **semantic** and **algorithmic** aspects.

Abstract semantics

Abstract semantic equation system

$$\mathcal{X}^\# : \mathcal{L} \rightarrow \mathcal{D}^\#$$

$$\mathcal{X}_\ell^\# \sqsupseteq \begin{cases} \top^\# & \text{if } \ell = e & \text{(entry)} \\ \bigcup_{(\ell', c, \ell) \in A} \mathbf{C}^\#[[c]] \mathcal{X}_{\ell'}^\# & \text{if } \ell \neq e & \text{(abstract transfer function)} \end{cases}$$

Soundness Theorem

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

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

$$\left\{ \begin{array}{ll} \mathcal{E} & \text{entry} \\ \mathcal{X}_\ell = \bigcup_{(\ell', c, \ell) \in A} \mathbf{C}[[c]] \mathcal{X}_{\ell'} & \text{if } \ell \neq e \end{array} \right.$$

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** ∇ to speed-up the convergence, if there are infinite strictly increasing chains in \mathcal{D}^\sharp .

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

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

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

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

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

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

Forward analysis:

$$\mathcal{X}_e^{\#0} \stackrel{\text{def}}{=} \top^{\#}$$

$$\mathcal{X}_{l \neq e}^{\#0} \stackrel{\text{def}}{=} \perp^{\#}$$

$$\mathcal{X}_l^{\#n+1} \stackrel{\text{def}}{=} \begin{cases} \top^{\#} & \text{if } l = e \\ \bigcup_{(\ell', c, \ell) \in A} C^{\#}[[c]] \mathcal{X}_{\ell'}^{\#n} & \text{if } l \notin \mathcal{W}, l \neq e \\ \mathcal{X}_l^{\#n} \nabla \bigcup_{(\ell', c, \ell) \in A} C^{\#}[[c]] \mathcal{X}_{\ell'}^{\#n} & \text{if } l \in \mathcal{W}, l \neq e \end{cases}$$

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

Abstract analysis (proof)

Proof of soundness:

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

If $\ell = e$, by definition: $\mathcal{X}_e^{\#\delta} = \top^\#$ and $\gamma(\top^\#) = \mathcal{E}$.

If $\ell \neq e$, $\ell \notin \mathcal{W}$, then $\mathcal{X}_\ell^{\#\delta} = \mathcal{X}_\ell^{\#\delta+1} = \bigcup_{(\ell', c, \ell) \in A}^\# \mathbf{C}^\# \llbracket c \rrbracket \mathcal{X}_{\ell'}^{\#\delta}$.

By soundness of $\bigcup^\#$ and $\mathbf{C}^\# \llbracket c \rrbracket$, $\gamma(\mathcal{X}_\ell^{\#\delta}) \supseteq \bigcup_{(\ell', c, \ell) \in A} \mathbf{C} \llbracket c \rrbracket \gamma(\mathcal{X}_{\ell'}^{\#\delta})$.

If $\ell \neq e$, $\ell \in \mathcal{W}$, then $\mathcal{X}_\ell^{\#\delta} = \mathcal{X}_\ell^{\#\delta+1} = \mathcal{X}_\ell^{\#\delta} \nabla \bigcup_{(\ell', c, \ell) \in A}^\# \mathbf{C}^\# \llbracket c \rrbracket \mathcal{X}_{\ell'}^{\#\delta}$.

By soundness of ∇ , $\gamma(\mathcal{X}_\ell^{\#\delta}) \supseteq \gamma(\bigcup_{(\ell', c, \ell) \in A}^\# \mathbf{C}^\# \llbracket c \rrbracket \mathcal{X}_{\ell'}^{\#\delta})$,

and so we also have $\gamma(\mathcal{X}_\ell^{\#\delta}) \supseteq \bigcup_{(\ell', c, \ell) \in A} \mathbf{C} \llbracket c \rrbracket \gamma(\mathcal{X}_{\ell'}^{\#\delta})$.

We have proved that $\lambda \ell. \gamma(\mathcal{X}_\ell^{\#\delta})$ 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 $\ell \in \mathcal{L}$, we denote by $i_\ell^1, \dots, i_\ell^k, \dots$ the increasing sequence of unstable indices, i.e., such that $\forall k, \mathcal{X}_\ell^{\#i_\ell^{k+1}} \neq \mathcal{X}_\ell^{\#i_\ell^k}$.

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

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

We argue that $\exists \ell \in \mathcal{W}$ such that $(i_\ell^k)_k$ is infinite as, otherwise,

$N = \max \{ i_\ell^k \mid \ell \in \mathcal{W} \} + |\mathcal{L}|$ is finite and satisfies: $\forall n \geq N, \forall \ell \in \mathcal{L}, \mathcal{X}_\ell^{\#n} = \mathcal{X}_\ell^{\#n+1}$, contradicting our assumption.

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

Then $\mathcal{Y}^{\#k+1} = \mathcal{Y}^{\#k} \nabla \mathcal{Z}^{\#k}$ for some sequence $\mathcal{Z}^{\#k}$.

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

Hence, the iteration must terminate in finite time.

Abstract analysis (cont.)

Backward refinement:

Given a forward analysis result $(\mathcal{X}_\ell^\#)_{\ell \in \mathcal{L}}$ and an abstract output $\mathcal{Y}^\#$ at x , we compute $(\mathcal{Y}_\ell^\#)_{\ell \in \mathcal{L}}$.

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

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

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

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

Non-relational domains

Value abstract domains

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

Numerical value abstract domain:

$\mathcal{B}^\#$	abstract values, machine-representable
$\gamma_b: \mathcal{B}^\# \rightarrow \mathcal{P}(\mathbb{I})$	concretization
\sqsubseteq_b	partial order
$\perp_b^\#, \top_b^\#$	represent \emptyset and \mathbb{I}
$\cup_b^\#, \cap_b^\#$	abstractions of \cup and \cap
∇_b	extrapolation operator
$\alpha_b: \mathcal{P}(\mathbb{I}) \rightarrow \mathcal{B}^\#$	abstraction (optional)

Derived abstract domain

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

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

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

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

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

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

Derived abstract domain (cont.)

$$\mathcal{X}^\# \sqsubseteq \mathcal{Y}^\# \stackrel{\text{def}}{\iff} \mathcal{X}^\# = \perp^\# \vee (\mathcal{X}^\#, \mathcal{Y}^\# \neq \perp^\# \wedge \forall V, \mathcal{X}^\#(V) \sqsubseteq_b \mathcal{Y}^\#(V))$$

$$\mathcal{X}^\# \cup^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} \begin{cases} \mathcal{Y}^\# & \text{if } \mathcal{X}^\# = \perp^\# \\ \mathcal{X}^\# & \text{if } \mathcal{Y}^\# = \perp^\# \\ \lambda V. \mathcal{X}^\#(V) \cup_b^\# \mathcal{Y}^\#(V) & \text{otherwise} \end{cases}$$

$$\mathcal{X}^\# \nabla \mathcal{Y}^\# \stackrel{\text{def}}{=} \begin{cases} \mathcal{Y}^\# & \text{if } \mathcal{X}^\# = \perp^\# \\ \mathcal{X}^\# & \text{if } \mathcal{Y}^\# = \perp^\# \\ \lambda V. \mathcal{X}^\#(V) \nabla_b \mathcal{Y}^\#(V) & \text{otherwise} \end{cases}$$

$$\mathcal{X}^\# \cap^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} \begin{cases} \perp^\# & \text{if } \mathcal{X}^\# = \perp^\# \text{ or } \mathcal{Y}^\# = \perp^\# \\ \perp^\# & \text{if } \exists V, \mathcal{X}^\#(V) \cap_b^\# \mathcal{Y}^\#(V) = \perp_b^\# \\ \lambda V. \mathcal{X}^\#(V) \cap_b^\# \mathcal{Y}^\#(V) & \text{otherwise} \end{cases}$$

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

- abstract operators $+_b^\#, \dots$ for $C^\#[[V \leftarrow e]]$
- backward abstract operators $\overleftarrow{+}_b^\#, \dots$
for $\overleftarrow{C}^\#[[V \leftarrow e]]$ and $C^\#[[e \bowtie 0]]^\#$

Cartesian abstraction

Non-relational domains “forget” all relationships between variables.

Cartesian abstraction:

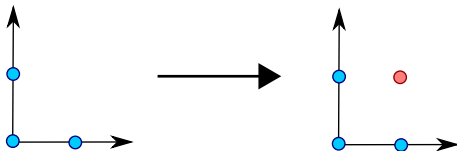
Upper closure operator $\rho_c : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{E})$

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

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



Generic non-relational abstract assignments

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

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

We can define:

- an abstract semantics of expressions: $E^\# \llbracket e \rrbracket : \mathcal{D}^\# \rightarrow \mathcal{B}^\#$

$$E^\# \llbracket e \rrbracket \perp^\# \stackrel{\text{def}}{=} \perp_b^\#$$

if $\mathcal{X}^\# \neq \perp^\#$:

$$E^\# \llbracket [c, c'] \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} [c, c']_b^\#$$

$$E^\# \llbracket V \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \mathcal{X}^\#(V)$$

$$E^\# \llbracket -e \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} -_b^\# E^\# \llbracket e \rrbracket \mathcal{X}^\#$$

$$E^\# \llbracket e_1 + e_2 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} E^\# \llbracket e_1 \rrbracket \mathcal{X}^\# +_b^\# E^\# \llbracket e_2 \rrbracket \mathcal{X}^\#$$

⋮

Generic non-relational abstract assignments (cont.)

We can then define:

- an abstract assignment:

$$C^\# \llbracket V \leftarrow e \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \begin{cases} \perp^\# & \text{if } \mathcal{V}_b^\# = \perp_b^\# \\ \mathcal{X}^\# [V \mapsto \mathcal{V}_b^\#] & \text{otherwise} \end{cases}$$

where $\mathcal{V}_b^\# = E^\# \llbracket e \rrbracket \mathcal{X}^\#$.

Using a Galois connection (α_b, γ_b) :

We can define **best** abstract arithmetic operators:

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

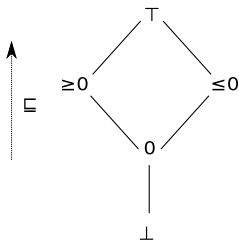
Note: in general, $E^\# \llbracket e \rrbracket$ is less precise than $\alpha_b \circ E \llbracket e \rrbracket \circ \gamma$

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

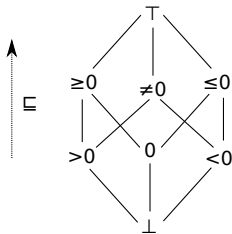
The sign domains

The sign lattices

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



simple signs



extended signs

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

The diagram implicitly defines $\cup^\#$ and $\cap^\#$ 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:

$$C^\# \llbracket X \leq 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \left(\begin{cases} \mathcal{X}^\# \llbracket X \mapsto 0 \rrbracket & \text{if } \mathcal{X}^\#(X) \in \{0, \geq 0\} \\ \mathcal{X}^\# \llbracket X \mapsto \leq 0 \rrbracket & \text{if } \mathcal{X}^\#(X) \in \{\top_b^\#, \leq 0\} \\ \perp^\# & \text{otherwise} \end{cases} \right)$$

$$C^\# \llbracket X - c \leq 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \left(\begin{cases} C^\# \llbracket X \leq 0 \rrbracket \mathcal{X}^\# & \text{if } c \leq 0 \\ \mathcal{X}^\# & \text{otherwise} \end{cases} \right)$$

$$C^\# \llbracket X - Y \leq 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} C^\# \llbracket X \leq 0 \rrbracket \mathcal{X}^\# & \text{if } \mathcal{X}^\#(Y) \in \{0, \leq 0\} \\ \mathcal{X}^\# & \text{otherwise} \end{array} \right. \cap^\# \left\{ \begin{array}{ll} C^\# \llbracket Y \geq 0 \rrbracket \mathcal{X}^\# & \text{if } \mathcal{X}^\#(X) \in \{0, \geq 0\} \\ \mathcal{X}^\# & \text{otherwise} \end{array} \right.$$

Other cases: $C^\# \llbracket expr \bowtie 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \mathcal{X}^\#$ is always a sound abstraction.

Simple sign analysis example

Example analysis using the simple sign domain:

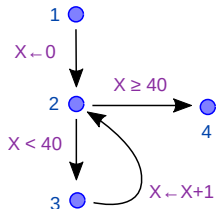
```

X ← 0;
while X < 40 do
  X ← X + 1
done
  
```

Program

$$\begin{cases}
 x_2^{\#i+1} &= C^{\#}[\![X \leftarrow 0]\!] x_1^{\#i} \cup \\
 & C^{\#}[\![X \leftarrow X + 1]\!] x_3^{\#i} \\
 x_3^{\#i+1} &= C^{\#}[\![X < 40]\!] x_2^{\#i} \\
 x_4^{\#i+1} &= C^{\#}[\![X \geq 40]\!] x_2^{\#i}
 \end{cases}$$

Iteration system



CFG

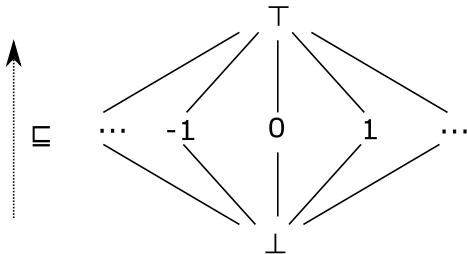
l	$x_l^{\#0}$	$x_l^{\#1}$	$x_l^{\#2}$	$x_l^{\#3}$	$x_l^{\#4}$	$x_l^{\#5}$
1	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$
2	$\perp^{\#}$	$X = 0$	$X = 0$	$X \geq 0$	$X \geq 0$	$X \geq 0$
3	$\perp^{\#}$	$\perp^{\#}$	$X = 0$	$X = 0$	$X \geq 0$	$X \geq 0$
4	$\perp^{\#}$	$\perp^{\#}$	$X = 0$	$X = 0$	$X \geq 0$	$X \geq 0$

Iterations

The constant domain

The constant lattice

Hasse diagram:



$$\mathcal{B}^\# = \perp \cup \{\top_b^\#, \perp_b^\#\}$$

The lattice is **flat** but **infinite**.

Operations on constants

Abstraction α : there is a Galois connection:

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

Derived abstract arithmetic operators:

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

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

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

Operations on constants (cont.)

Abstract test examples:

$$C^\# \llbracket X - c = 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \begin{cases} \perp^\# & \text{if } \mathcal{X}^\#(X) \notin \{c, \top_b^\#\} \\ \mathcal{X}^\#[X \mapsto c] & \text{otherwise} \end{cases}$$

$$C^\# \llbracket X - Y - c = 0 \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \left(\begin{array}{l} \left(\begin{array}{l} C^\# \llbracket X - (\mathcal{X}^\#(Y) + c) = 0 \rrbracket \mathcal{X}^\# \\ \mathcal{X}^\# \end{array} \right) \text{ if } \mathcal{X}^\#(Y) \notin \{\perp_b^\#, \top_b^\#\} \\ \text{otherwise} \end{array} \right) \cap^\# \left(\begin{array}{l} \left(\begin{array}{l} C^\# \llbracket Y - (\mathcal{X}^\#(X) - c) = 0 \rrbracket \mathcal{X}^\# \\ \mathcal{X}^\# \end{array} \right) \text{ if } \mathcal{X}^\#(X) \notin \{\perp_b^\#, \top_b^\#\} \\ \text{otherwise} \end{array} \right)$$

Constant analysis example

B^\sharp has **finite height**, the $(\mathcal{X}_\ell^{\sharp i})$ **converge in finite time**.
(even though 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: $\begin{cases} X = T_b^\sharp \\ Y = 7 \end{cases}$

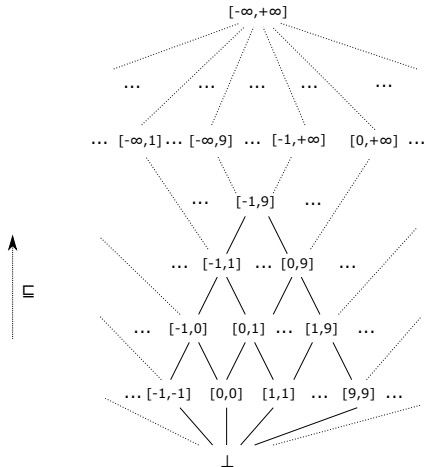
Note: the analysis can find constants **that do not appear syntactically** in the program.

The interval domain

The interval lattice

Introduced by [Cous76].

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



Note: intervals are open at infinite bounds $+\infty$, $-\infty$.

The interval lattice (cont.)

Galois connection (α_b, γ_b) :

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

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

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

Partial order:

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

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

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

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

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

Interval abstract arithmetic operators

$$[c, c'] \#_b \stackrel{\text{def}}{=} [c, c']$$

$$-\#_b [a, b] \stackrel{\text{def}}{=} [-b, -a]$$

$$[a, b] \#_b [c, d] \stackrel{\text{def}}{=} [a + c, b + d]$$

$$[a, b] -\#_b [c, d] \stackrel{\text{def}}{=} [a - d, b - c]$$

$$[a, b] \times\#_b [c, d] \stackrel{\text{def}}{=} [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

$$[a, b] /\#_b [c, d] \stackrel{\text{def}}{=} \begin{cases} \perp_b & \text{if } c = d = 0 \\ [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)] & \text{else if } 0 \leq c \\ [-b, -a] /\#_b [-d, -c] & \text{else if } d \leq 0 \\ ([a, b] /\#_b [c, 0]) \cup_b ([a, b] /\#_b [0, d]) & \text{otherwise} \end{cases}$$

where $\begin{cases} \pm\infty \times 0 = 0, & 0/0 = 0, & \forall x, x/\pm\infty = 0 \\ \forall x > 0, x/0 = +\infty, & \forall x < 0, x/0 = -\infty \end{cases}$

Operators are **strict**: $-\#_b \perp_b = \perp_b$, $[a, b] \#_b \perp_b = \perp_b$, etc.

Exactness and optimality: Example proofs

Proof: exactness of $+_b^\#$

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

Proof optimality of $\cup_b^\#$

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

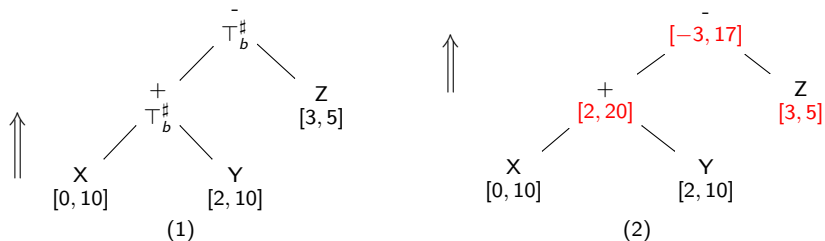
but $\cup_b^\#$ is not exact

...

Generic interval abstract tests, step 1

Example: $C^\# \llbracket X + Y - Z \leq 0 \rrbracket \mathcal{X}^\#$
 with $\mathcal{X}^\# = \{ X \mapsto [0, 10], Y \mapsto [2, 10], Z \mapsto [3, 5] \}$

First step: **annotate** the expression tree with intervals

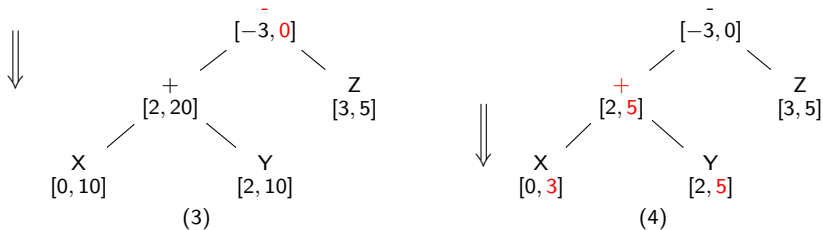


Bottom-up evaluation similar to interval expression evaluation using $+_b^\#$, $-_b^\#$, etc. but storing intervals at each node.

Generic interval abstract tests, step 2

Example: $C^\# \llbracket X + Y - Z \leq 0 \rrbracket \mathcal{X}^\#$
 with $\mathcal{X}^\# = \{ X \mapsto [0, 10], Y \mapsto [2, 10], Z \mapsto [3, 5] \}$

Second step: **top-down** expression **refinement**.



- refine the **root** interval, knowing that the result should be negative;
- **propagate** refined intervals **downwards**;
- intervals at **leaf variables** provide new information to store into $\mathcal{X}^\#$.
 $\{ X \mapsto [0, 3], Y \mapsto [2, 5], Z \mapsto [3, 5] \}$

Backward arithmetic and comparison operators

In general, we need **sound backward** arithmetic and comparison operators that **refine** their arguments given a result.

Soundness condition: for $\leq_b^\#, \nrightarrow_b^\#, \leftarrow_b^\#, \dots$

$$\begin{aligned} \mathcal{X}_b^{\#'} = \leq_b^\#(\mathcal{X}_b^\#) &\implies \\ \{x \in \gamma_b(\mathcal{X}_b^\#) \mid x \leq 0\} &\subseteq \gamma_b(\mathcal{X}_b^{\#'}) \subseteq \gamma_b(\mathcal{X}_b^\#) \end{aligned}$$

$$\begin{aligned} \mathcal{X}_b^{\#'} = \leftarrow_b^\#(\mathcal{X}_b^\#, \mathcal{R}_b^\#) &\implies \\ \{x \mid x \in \gamma_b(\mathcal{X}_b^\#), -x \in \gamma_b(\mathcal{R}_b^\#)\} &\subseteq \gamma_b(\mathcal{X}_b^{\#'}) \subseteq \gamma_b(\mathcal{X}_b^\#) \end{aligned}$$

$$\begin{aligned} (\mathcal{X}_b^{\#'}, \mathcal{Y}_b^{\#'}) = \nrightarrow_b^\#(\mathcal{X}_b^\#, \mathcal{Y}_b^\#, \mathcal{R}_b^\#) &\implies \\ \{x \in \gamma_b(\mathcal{X}_b^\#) \mid \exists y \in \gamma_b(\mathcal{Y}_b^\#), x + y \in \gamma_b(\mathcal{R}_b^\#)\} &\subseteq \gamma_b(\mathcal{X}_b^{\#'}) \subseteq \gamma_b(\mathcal{X}_b^\#) \\ \{y \in \gamma_b(\mathcal{Y}_b^\#) \mid \exists x \in \gamma_b(\mathcal{X}_b^\#), x + y \in \gamma_b(\mathcal{R}_b^\#)\} &\subseteq \gamma_b(\mathcal{Y}_b^{\#'}) \subseteq \gamma_b(\mathcal{Y}_b^\#) \end{aligned}$$

⋮

Note: **best** backward operators can be designed with α_b :

e.g. for $\nrightarrow_b^\#$: $\mathcal{X}_b^{\#'} = \alpha_b(\{x \in \gamma_b(\mathcal{X}_b^\#) \mid \exists y \in \gamma_b(\mathcal{Y}_b^\#), x + y \in \gamma_b(\mathcal{R}_b^\#)\})$

Generic backward operator construction

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

$$\overleftarrow{0}_b^\#(x_b^\#) \stackrel{\text{def}}{=} x_b^\# \cap_b^\# [-\infty, 0]_b^\#$$

$$\overleftarrow{_}{}_b^\#(x_b^\#, \mathcal{R}_b^\#) \stackrel{\text{def}}{=} x_b^\# \cap_b^\# (-_{}_b^\# \mathcal{R}_b^\#)$$

$$\overleftarrow{+}{}_b^\#(x_b^\#, y_b^\#, \mathcal{R}_b^\#) \stackrel{\text{def}}{=} (x_b^\# \cap_b^\# (\mathcal{R}_b^\# -_{}_b^\# y_b^\#), y_b^\# \cap_b^\# (\mathcal{R}_b^\# -_{}_b^\# x_b^\#))$$

$$\overleftarrow{+}{}_b^\#(x_b^\#, y_b^\#, \mathcal{R}_b^\#) \stackrel{\text{def}}{=} (x_b^\# \cap_b^\# (\mathcal{R}_b^\# +_{}_b^\# y_b^\#), y_b^\# \cap_b^\# (x_b^\# -_{}_b^\# \mathcal{R}_b^\#))$$

$$\overleftarrow{/}{}_b^\#(x_b^\#, y_b^\#, \mathcal{R}_b^\#) \stackrel{\text{def}}{=} (x_b^\# \cap_b^\# (\mathcal{R}_b^\# /_{}_b^\# y_b^\#), y_b^\# \cap_b^\# (\mathcal{R}_b^\# /_{}_b^\# x_b^\#))$$

$$\overleftarrow{\uparrow}{}_b^\#(x_b^\#, y_b^\#, \mathcal{R}_b^\#) \stackrel{\text{def}}{=} (x_b^\# \cap_b^\# (\mathcal{S}_b^\# \times_{}_b^\# y_b^\#), y_b^\# \cap_b^\# ((x_b^\# /_{}_b^\# \mathcal{S}_b^\#) \cup_b^\# [0, 0]_b^\#))$$

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

Note: $\overleftarrow{\diamond}{}_b^\#(x_b^\#, y_b^\#, \mathcal{R}_b^\#) = (x_b^\#, y_b^\#)$ is always sound (no refinement).

Interval backward operators

Applying the generic construction to the interval domain:

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

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

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

...

Generic non-relational backward assignment

Abstract function: $\overleftarrow{C}^\# \llbracket V \leftarrow e \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#)$

over-approximates $\gamma(\mathcal{X}^\#) \cap \overleftarrow{C} \llbracket V \leftarrow e \rrbracket \gamma(\mathcal{R}^\#)$ given:

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

Algorithm: similar to the abstract test

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

Note:

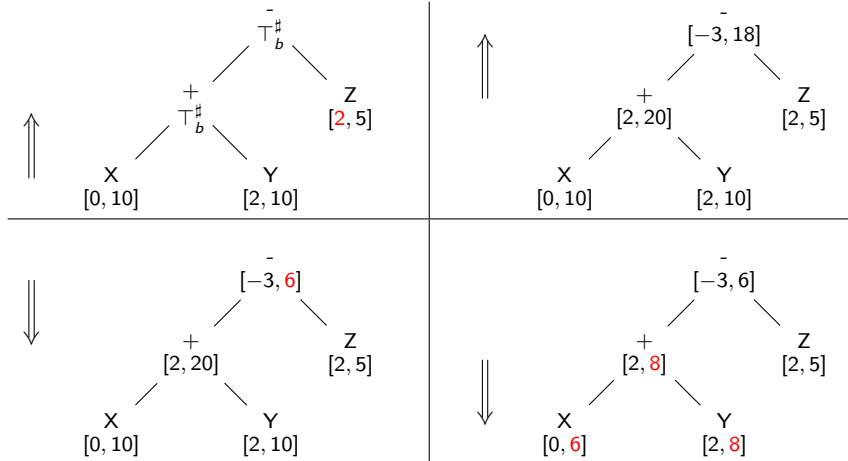
- local iterations can also be used
- fallback: $\overleftarrow{C}^\# \llbracket V \leftarrow e \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#) = \mathcal{X}^\# \cap^\# (\mathcal{R}^\# [V \mapsto T_b^\#])$

Interval backward assignment example

Example: $\overleftarrow{C}^\# \llbracket X \leftarrow X + Y - Z \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#)$

with $\mathcal{X}^\# = \{ X \mapsto [0, 10], Y \mapsto [2, 10], Z \mapsto [1, 5] \}$

and $\mathcal{R}^\# = \{ X \mapsto [-6, 6], Y \mapsto [2, 10], Z \mapsto [2, 6] \}$



Interval widening

Widening on non-relational domains:

Given a value widening $\nabla_b: \mathcal{B}^\# \times \mathcal{B}^\# \rightarrow \mathcal{B}^\#$,

we extend it point-wisely into a widening $\nabla: \mathcal{D}^\# \times \mathcal{D}^\# \rightarrow \mathcal{D}^\#$:

$$\mathcal{X}^\# \nabla \mathcal{Y}^\# \stackrel{\text{def}}{=} \lambda V. (\mathcal{X}^\#(V) \nabla_b \mathcal{Y}^\#(V))$$

Interval widening example:

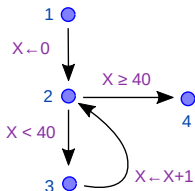
$$\perp^\# \nabla_b \mathcal{X}^\# \stackrel{\text{def}}{=} \mathcal{X}^\#$$

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

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

Analysis with widening example

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



ℓ	$x_\ell^{\#0}$	$x_\ell^{\#1}$	$x_\ell^{\#2}$	$x_\ell^{\#3}$	$x_\ell^{\#4}$	$x_\ell^{\#5}$
1	$\top^\#$	$\top^\#$	$\top^\#$	$\top^\#$	$\top^\#$	$\top^\#$
2 ∇	$\perp^\#$	$= 0$	$= 0$	≥ 0	≥ 0	≥ 0
3	$\perp^\#$	$\perp^\#$	$= 0$	$= 0$	$\in [0, 39]$	$\in [0, 39]$
4	$\perp^\#$	$\perp^\#$	$\perp^\#$	$\perp^\#$	≥ 40	≥ 40

More precisely, at the widening point:

$$\begin{aligned}
 x_2^{\#1} &= \perp^\# & \nabla_b ([0, 0] \cup_b \perp^\#) &= \perp^\# & \nabla_b [0, 0] &= [0, 0] \\
 x_2^{\#2} &= [0, 0] & \nabla_b ([0, 0] \cup_b \perp^\#) &= [0, 0] & \nabla_b [0, 0] &= [0, 0] \\
 x_2^{\#3} &= [0, 0] & \nabla_b ([0, 0] \cup_b [1, 1]) &= [0, 0] & \nabla_b [0, 1] &= [0, +\infty[\\
 x_2^{\#4} &= [0, +\infty] & \nabla_b ([0, 0] \cup_b [1, 40]) &= [0, +\infty] & \nabla_b [0, 40] &= [0, +\infty]
 \end{aligned}$$

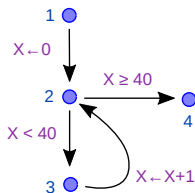
Note that the most precise interval abstraction would be

$X \in [0, 40]$ at 2, and $X = 40$ 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	$x_l^{\#1}$	$x_l^{\#2}$	$x_l^{\#3}$	$x_l^{\#4}$	$x_l^{\#5}$	$x_l^{\#6}$
1	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$
2	$= 0$	$= 0$	$\in [0, 1]$	$\in [0, 1]$	≥ 0	≥ 0
3 ∇	$\perp^{\#}$	$= 0$	$= 0$	≥ 0	≥ 0	≥ 0
4	$\perp^{\#}$	$\perp^{\#}$	$\perp^{\#}$	$\perp^{\#}$	$\perp^{\#}$	≥ 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** Δ .

Definition: narrowing Δ

Binary operator $\mathcal{D}^\# \times \mathcal{D}^\# \rightarrow \mathcal{D}^\#$ such that:

- $(\mathcal{X}^\# \cap^\# \mathcal{Y}^\#) \sqsubseteq (\mathcal{X}^\# \Delta \mathcal{Y}^\#) \sqsubseteq \mathcal{X}^\#$,
- for all sequences $(\mathcal{X}_i^\#)$, the decreasing sequence $(\mathcal{Y}_i^\#)$

defined by
$$\begin{cases} \mathcal{Y}_0^\# & \stackrel{\text{def}}{=} & \mathcal{X}_0^\# \\ \mathcal{Y}_{i+1}^\# & \stackrel{\text{def}}{=} & \mathcal{Y}_i^\# \Delta \mathcal{X}_{i+1}^\# \end{cases}$$
 is **stationary**.

This is not the dual of a widening!

Narrowing examples

Trivial narrowing:

$\mathcal{X}^\# \Delta \mathcal{Y}^\# \stackrel{\text{def}}{=} \mathcal{X}^\#$ is a correct narrowing.

Finite-time intersection narrowing:

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

Interval narrowing:

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

(refine only infinite bounds)

Point-wise extension to $\mathcal{D}^\#$: $\mathcal{X}^\# \Delta \mathcal{Y}^\# \stackrel{\text{def}}{=} \lambda V. (\mathcal{X}^\#(V) \Delta_b \mathcal{Y}^\#(V))$

Iterations with narrowing

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

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

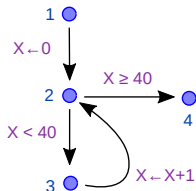
The following sequence is computed:

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

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

Analysis with narrowing example

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



l	$\mathcal{Y}_l^{\#0}$	$\mathcal{Y}_l^{\#1}$	$\mathcal{Y}_l^{\#2}$	$\mathcal{Y}_l^{\#3}$
1	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$	$\top^{\#}$
2 Δ	≥ 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:

$$\mathcal{Y}_2^{\#1} = [0, +\infty] \Delta_b ([0, 0] \cup_b [1, 40]) = [0, +\infty] \Delta_b [0, 40] = [0, 40]$$

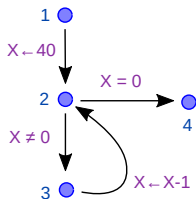
$$\mathcal{Y}_2^{\#2} = [0, 40] \Delta_b ([0, 0] \cup_b [1, 40]) = [0, 40] \Delta_b [0, 40] = [0, 40]$$

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

We found the most precise invariants!

Improving the widening

Example of imprecise analysis



ℓ	intervals with ∇_b	extended signs	intervals with ∇'_b
1	$\top^\#$	$\top^\#$	$\top^\#$
2 ∇	$X \leq 40$	$X \geq 0$	$X \in [0, 40]$
3	$X \leq 40$	$X > 0$	$X \in [0, 40]$
4	$X = 0$	$X = 0$	$X = 0$

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

Solution: improve the interval widening

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

(∇'_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 $X \in [0, +\infty]$.

$$\mathcal{X}_{\bullet}^{\#} = [0, 0] \nabla_b ([0, 0] \cup^{\#} [0, 1]) = [0, 0] \nabla_b [0, 1] = [0, +\infty[$$

- Narrowing is unable to refine the invariant:

$$\mathcal{Y}_{\bullet}^{\#} = [0, +\infty] \Delta_b ([0, 0] \cup^{\#} [0, +\infty]) = [0, +\infty[$$

(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 ∇_b^T

$$[a, b] \nabla_b^T [c, d] \stackrel{\text{def}}{=} \left[\begin{array}{ll} \left\{ \begin{array}{ll} a & \text{if } a \leq c \\ \max \{x \in T \mid x \leq c\} & \text{otherwise} \end{array} \right. & , \\ \left. \begin{array}{ll} b & \text{if } b \geq d \\ \min \{x \in T \mid x \geq d\} & \text{otherwise} \end{array} \right. \end{array} \right]$$

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

Widening with thresholds (cont.)

Applications:

- On the previous example, we find:
 $X \in [0, \min \{x \in T \mid x \geq 40\}]$.
- 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
```

15 is stable

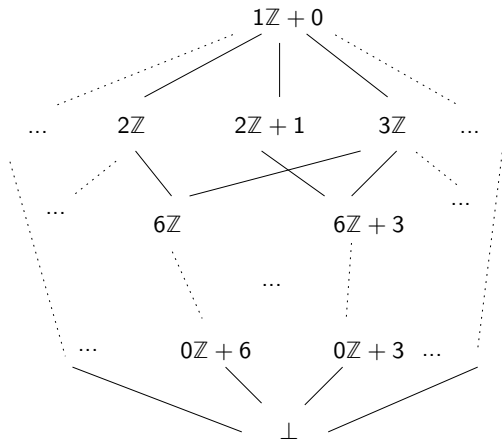
```
while 1 = 1 do
  X ← X + 1;
  if X ≠ 10 then X ← 0 fi
done
```

no stable bound

The congruence domain

The congruence lattice

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



Introduced by Granger [Gran89].

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

The congruence lattice (cont.)

Concretization:

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

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:

- $y/y' \stackrel{\text{def}}{\iff} y \text{ divides } y' (\exists k \in \mathbb{N}, y' = ky)$ (note that $\forall y: y/0$)
- $x \equiv x' [y] \stackrel{\text{def}}{\iff} y \mid |x - x'|$ (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 $\mathcal{B}^\#$:

- $(a\mathbb{Z} + b) \sqsubseteq_b (a'\mathbb{Z} + b') \stackrel{\text{def}}{\iff} a'/a \text{ and } b \equiv b' [a']$
- $\top_b \stackrel{\text{def}}{=} (1\mathbb{Z} + 0)$
- $(a\mathbb{Z} + b) \cup_b^\# (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} (a \wedge a' \wedge |b - b'|)\mathbb{Z} + b$
- $(a\mathbb{Z} + b) \cap_b^\# (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} \begin{cases} (a \vee a')\mathbb{Z} + b'' & \text{if } b \equiv b' [a \wedge a'] \\ \perp_b^\# & \text{otherwise} \end{cases}$
 b'' such that $b'' \equiv b [a \vee a'] \equiv b' [a \vee a']$ is given by Bezout's Theorem.

Galois connection: $\alpha_b(\mathcal{X}) = \bigcup_{c \in \mathcal{X}}^\# (0\mathbb{Z} + c)$

(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:

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

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

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

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

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

$$(a\mathbb{Z} + b) /_b^\# (a'\mathbb{Z} + b') \stackrel{\text{def}}{=} \begin{cases} \perp_b^\# & \text{if } a'\mathbb{Z} + b' = 0\mathbb{Z} + 0 \\ (a/|b'|)\mathbb{Z} + (b/b') & \text{if } a' = 0, b' \neq 0, b'|a, \text{ and } b'|b \\ \top_b^\# & \text{otherwise (not optimal)} \end{cases}$$

Abstract congruence operators (cont.)

Test operators:

$$\begin{aligned} \overleftarrow{\leq}_b^\# (a\mathbb{Z} + b) &\stackrel{\text{def}}{=} \begin{cases} \perp_b^\# & \text{if } a = 0, b > 0 \\ a\mathbb{Z} + b & \text{otherwise} \end{cases} \\ &\vdots \end{aligned}$$

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

Extrapolation operators:

- no infinite increasing chain \implies no need for ∇
- infinite decreasing chains $\implies \Delta$ needed

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

Note: $\mathcal{X}^\# \Delta \mathcal{Y}^\# \stackrel{\text{def}}{=} \mathcal{X}^\#$ 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 $\left\{ \begin{array}{l} X \in 2\mathbb{Z} \\ Y \in 6\mathbb{Z} + 2 \end{array} \right.$

Reduced products

Non-reduced product of domains

Product representation:

Cartesian product $\mathcal{D}_{1 \times 2}^\#$ of $\mathcal{D}_1^\#$ and $\mathcal{D}_2^\#$:

- $\mathcal{D}_{1 \times 2}^\# \stackrel{\text{def}}{=} \mathcal{D}_1^\# \times \mathcal{D}_2^\#$
- $\gamma_{1 \times 2}(\mathcal{X}_1^\#, \mathcal{X}_2^\#) \stackrel{\text{def}}{=} \gamma_1(\mathcal{X}_1^\#) \cap \gamma_2(\mathcal{X}_2^\#)$
- $\alpha_{1 \times 2}(\mathcal{X}) \stackrel{\text{def}}{=} (\alpha_1(\mathcal{X}), \alpha_2(\mathcal{X}))$
- $(\mathcal{X}_1^\#, \mathcal{X}_2^\#) \sqsubseteq_{1 \times 2} (\mathcal{Y}_1^\#, \mathcal{Y}_2^\#) \iff \mathcal{X}_1^\# \sqsubseteq_1 \mathcal{Y}_1^\# \text{ and } \mathcal{X}_2^\# \sqsubseteq_2 \mathcal{Y}_2^\#$

Abstract operators: performed in parallel on both components:

- $(\mathcal{X}_1^\#, \mathcal{X}_2^\#) \cup_{1 \times 2}^\# (\mathcal{Y}_1^\#, \mathcal{Y}_2^\#) \stackrel{\text{def}}{=} (\mathcal{X}_1^\# \cup_1^\# \mathcal{Y}_1^\#, \mathcal{X}_2^\# \cup_2^\# \mathcal{Y}_2^\#)$
- $(\mathcal{X}_1^\#, \mathcal{X}_2^\#) \nabla_{1 \times 2} (\mathcal{Y}_1^\#, \mathcal{Y}_2^\#) \stackrel{\text{def}}{=} (\mathcal{X}_1^\# \nabla_1 \mathcal{Y}_1^\#, \mathcal{X}_2^\# \nabla_2 \mathcal{Y}_2^\#)$
- $\mathbf{C}^\# \llbracket c \rrbracket_{1 \times 2} (\mathcal{X}_1^\#, \mathcal{X}_2^\#) \stackrel{\text{def}}{=} (\mathbf{C}^\# \llbracket c \rrbracket_1 (\mathcal{X}_1^\#), \mathbf{C}^\# \llbracket c \rrbracket_2 (\mathcal{X}_2^\#))$

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
•	$X \in [11, 12]$	$X \equiv 1 [2]$	$X = 11$
♦	$X = 12$	$X \equiv 1 [2]$	\emptyset
★	$X = 0$	$X = 0$	$X = 0$

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:

$$\rho : \mathcal{D}_{1 \times 2}^\sharp \rightarrow \mathcal{D}_{1 \times 2}^\sharp$$

$$\rho(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \stackrel{\text{def}}{=} (\alpha_1(\gamma_1(\mathcal{X}_1^\sharp) \cap \gamma_2(\mathcal{X}_2^\sharp)), \alpha_2(\gamma_1(\mathcal{X}_1^\sharp) \cap \gamma_2(\mathcal{X}_2^\sharp)))$$

ρ propagates information between domains.

Application:

We can reduce the result of each abstract operator, except ∇ :

- $(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \cup_{1 \times 2}^\sharp (\mathcal{Y}_1^\sharp, \mathcal{Y}_2^\sharp) \stackrel{\text{def}}{=} \rho(\mathcal{X}_1^\sharp \cup_1^\sharp \mathcal{Y}_1^\sharp, \mathcal{X}_2^\sharp \cup_2^\sharp \mathcal{Y}_2^\sharp),$
- $\mathbf{C}^\sharp \llbracket c \rrbracket_{1 \times 2}(\mathcal{X}_1^\sharp, \mathcal{X}_2^\sharp) \stackrel{\text{def}}{=} \rho(\mathbf{C}^\sharp \llbracket c \rrbracket_1(\mathcal{X}_1^\sharp), \mathbf{C}^\sharp \llbracket c \rrbracket_2(\mathcal{X}_2^\sharp)).$

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

Fully-reduced product example

Reduction example: between the **interval** and **congruence** domains:

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

$$b' \stackrel{\text{def}}{=} \max \{ x \leq b \mid x \equiv d [c] \}$$

We get:

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

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

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)
- $\rho_b([1, 3], 4\mathbb{Z}) = (\perp_b^\#, \perp_b^\#)$

Partially-reduced product

Definition: of a **partial** reduction:

any function $\rho : \mathcal{D}_{1 \times 2}^\# \rightarrow \mathcal{D}_{1 \times 2}^\#$ such that:

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

Useful when:

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

Partial reduction example:

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

(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. FSTTSC'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.