Static analysis of run-time errors in programs with floating-point computations

Antoine Miné

LIP6, Sorbonne Universités, UPMC
Paris, France

Pequan Seminar
10 December 2015

The cost of software failure

- Patriot MIM-104 failure, 25 February 1991
  (death of 28 soldiers[1])
- Ariane 5 failure, 4 June 1996
  (cost estimated at more than 370 000 000 US$[2])
- Toyota electronic throttle control system failure, 2005
  (at least 89 death[3])
- Heartbleed bug in OpenSSL, April 2014
- Stagefright bug in Android, Summer 2015
  (multiple array overflows in 900 million devices, some exploitable)
- economic cost of software bugs is tremendous[4]

---

[1] R. Skeel. "Roundoff Error and the Patriot Missile". SIAM News, volume 25, nr 4.

[2] M. Dowson. "The Ariane 5 Software Failure". Software Engineering Notes 22 (2): 84, March 1997.

[3] CBSNews. Toyota "Unintended Acceleration" Has Killed 89. 20 March 2014.

[4] NIST. Software errors cost U.S. economy $59.5 billion annually. Tech. report, NIST Planning Report, 2002.

Static analysis

Goal: static analysis [CousotCousot-ISP76]

Static (automatic) discovery
of **dynamic** (semantic) properties of programs.

Applications:

- compilation and optimisation, e.g.:
  - array bound check elimination
  - alias analysis
- verification, e.g.:
  - infer invariants
  - prove the absence of run-time errors
    (division by zero, overflow, invalid array access)
  - prove functional properties

We focus here on numerical properties of numerical variables.

Example analysis: inferring numeric invariants

Insertion Sort

```
for i=1 to 99 do
  p := T[i]; j := i+1;
  while j <= 100 and T[j] < p do
    T[j-1] := T[j]; j := j+1;
  end;
  T[j-1] := p;
end;
```

Example analysis: inferring numeric invariants

Interval analysis:

Insertion Sort

```
for i=1 to 99 do
    i ∈ [1, 99]
  p := T[i]; j := i+1;
    i ∈ [1, 99], j ∈ [2, 100]
  while j <= 100 and T[j] < p do
      i ∈ [1, 99], j ∈ [2, 100]
    T[j-1] := T[j]; j := j+1;
      i ∈ [1, 99], j ∈ [3, 101]
  end;
    i ∈ [1, 99], j ∈ [2, 101]
  T[j-1] := p;
end;
```

$\Longrightarrow$ there is no out of bound array access

Example analysis: inferring numeric invariants

Linear inequality analysis:

Insertion Sort

```
for i=1 to 99 do
    i ∈ [1,99]
  p := T[i]; j := i+1;
    i ∈ [1,99], j = i+1
  while j <= 100 and T[j] < p do
      i ∈ [1,99], i+1 ≤ j ≤ 100
    T[j-1] := T[j]; j := j+1;
      i ∈ [1,99], i+2 ≤ j ≤ 101
  end;
    i ∈ [1,99], i+1 ≤ j ≤ 101
  T[j-1] := p;
end;
```

Theoretical background

Abstract interpretation: unifying theory of program semantics

[CousotCousot-POPL77]

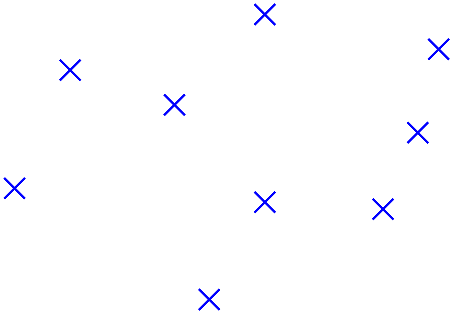
Provide theoretical tools to design and compare static analyses that:

- always terminate
- are approximate (solve undecidability and efficiency issues)
- are sound by construction (no behavior is omitted)

Analysis design roadmap:

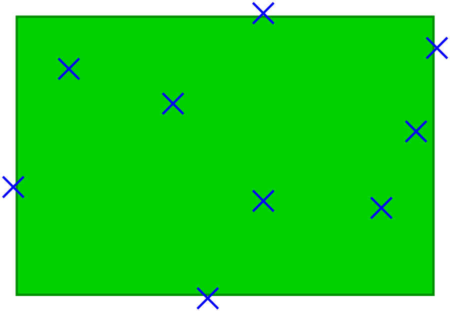
1. concrete semantics
2. abstract domains

Abstract domain examples



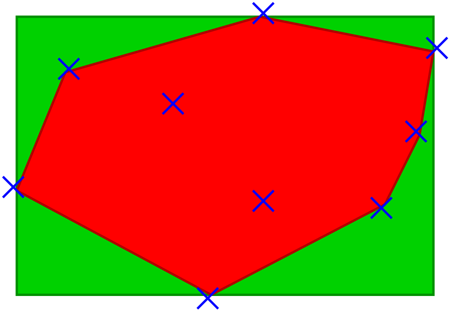
concrete $\mathcal{D}$: $\{(0,3),(5.5,0),(12,7),\ldots\}$ (not computable)

Abstract domain examples



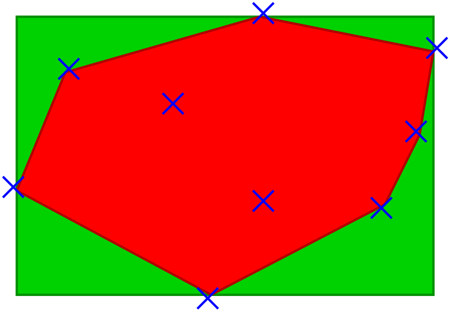
concrete $\mathcal{D}$:	$\{(0,3),(5.5,0),(12,7),\ldots\}$	(not computable)
boxes $\mathcal{D}^\sharp_b$:	$X \in [0;12] \wedge Y \in [0;8]$	(linear cost)

Abstract domain examples



concrete $\mathcal{D}$:	$\{(0,3),(5.5,0),(12,7),\ldots\}$	(not computable)
boxes $\mathcal{D}^\sharp_b$:	$X \in [0;12] \wedge Y \in [0;8]$	(linear cost)
polyhedra $\mathcal{D}^\sharp_p$:	$6X + 11Y \geq 33 \wedge \cdots$	(exponential cost)

Abstract domain examples



concrete $\mathcal{D}$:	$\{(0,3),(5.5,0),(12,7),\ldots\}$	(not computable)
boxes $\mathcal{D}_b^\sharp$:	$X \in [0;12] \wedge Y \in [0;8]$	(linear cost)
polyhedra $\mathcal{D}_p^\sharp$:	$6X + 11Y \geq 33 \wedge \cdots$	(exponential cost)

$\Longrightarrow$ trade-off cost vs. precision and expressiveness.

Overview

- **Classic domains** (on rationals)
  - concrete semantics
  - interval domain
  - polyhedra domain

- **Floating-point domains**
  - concrete semantics of floats
  - floats intervals
  - linearization of float expressions
  - Application: the Astrée analyzer
  - float polyhedra

Classic Domains

Toy language: syntax

arithmetic expressions:

exp	::=	V	variable $\mathtt{V} \in \mathcal{V}$
	|	$-$exp	negation
	|	exp $\diamond$ exp	binary operation: $\diamond \in \{\,+, -, \times, /\,\}$
	|	$[c, c']$	constant range, $c, c' \in \mathbb{Q} \cup \{\,\pm\infty\,\}$
			(c is a shorthand for $[c, c]$)

programs:

prog	::=	V := exp	assignment
	|	if exp $\bowtie$ 0 then prog else prog fi	test
	|	while exp $\bowtie$ 0 do prog done	loop
	|	prog; prog	sequence

Finite set $\mathcal{V}$ of variables, with value in $\mathbb{Q}$
(later extended to floats $\mathbb{F}$ and machine integers $\mathbb{M}$)

Concrete semantics

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

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

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

Concrete semantics

Semantics of programs: $\mathrm{C}[\![\, p \,]\!] : \mathcal{D} \rightarrow \mathcal{D}$

where $\mathcal{D} \stackrel{\text{def}}{=} \mathcal{P}(\mathcal{V} \rightarrow \mathbb{Q})$

A transfer function for p defines a relation on environments $\rho \in \mathcal{D}$:

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

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

$$\mathrm{C}[\![\, b_1; b_2 \,]\!]\mathcal{X} \stackrel{\text{def}}{=} \mathrm{C}[\![\, b_2 \,]\!](\mathrm{C}[\![\, b_1 \,]\!]\mathcal{X})$$

$$\mathrm{C}[\![\, \mathtt{if}\ e \bowtie 0\ \mathtt{then}\ b_1\ \mathtt{else}\ b_2 \,]\!]\mathcal{X} \stackrel{\text{def}}{=} (\mathrm{C}[\![\, b_1 \,]\!] \circ \mathrm{C}[\![\, e \bowtie 0 \,]\!])\mathcal{X} \cup (\mathrm{C}[\![\, b_2 \,]\!] \circ \mathrm{C}[\![\, e \not\bowtie 0 \,]\!])\mathcal{X}$$

$$\mathrm{C}[\![\, \mathtt{while}\ e \bowtie 0\ \mathtt{do}\ b\ \mathtt{done} \,]\!]\mathcal{X} \stackrel{\text{def}}{=} \mathrm{C}[\![\, e \not\bowtie 0 \,]\!](\mathrm{lfp}\,\lambda\mathcal{Y}.\mathcal{X} \cup (\mathrm{C}[\![\, b \,]\!] \circ \mathrm{C}[\![\, e \bowtie 0 \,]\!])\mathcal{Y})$$

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

Abstract domains

- Abstract elements:
  - $\mathcal{D}^\sharp$ set of computer-representable elements
  - $\gamma : \mathcal{D}^\sharp \to \mathcal{D}$ concretization
  - $\subseteq^\sharp$ approximation order: $\mathcal{X}^\sharp \subseteq^\sharp \mathcal{Y}^\sharp \implies \gamma(\mathcal{X}^\sharp) \subseteq \gamma(\mathcal{Y}^\sharp)$
- Abstract operators:
  - $\mathrm{C}^\sharp[\![c]\!] : \mathcal{D}^\sharp \to \mathcal{D}^\sharp$ and $\cup^\sharp : (\mathcal{D}^\sharp \times \mathcal{D}^\sharp) \to \mathcal{D}^\sharp$
  - soundness: $F \circ \gamma \subseteq \gamma \circ F^\sharp$
- Fixpoint extrapolation
  - $\triangledown : (\mathcal{D}^\sharp \times \mathcal{D}^\sharp) \to \mathcal{D}^\sharp$ widening
  - soundness: $\gamma(\mathcal{X}^\sharp) \cup \gamma(\mathcal{Y}^\sharp) \subseteq \gamma(\mathcal{X}^\sharp \triangledown \mathcal{Y}^\sharp)$
  - 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: $\exists n < \omega, \mathcal{X}_{n+1}^\sharp = \mathcal{X}_n^\sharp$

Both semantics and algorithmic aspects.

Abstract semantics

Given by the abstract domain:

- sound $\mathrm{C}^\sharp[\![\, \mathtt{V} := e \,]\!]$, $\mathrm{C}^\sharp[\![\, e \bowtie 0 \,]\!]$, $\cup^\sharp$
- sound and terminating $\triangledown$

Derived analysis: from the concrete...

$$\mathrm{C}[\![\, b_1; b_2 \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=} \mathrm{C}[\![\, b_2 \,]\!]\,(\mathrm{C}[\![\, b_1 \,]\!]\,\mathcal{X})$$

$$\mathrm{C}[\![\, \mathtt{if}\ e \bowtie 0\ \mathtt{then}\ b_1\ \mathtt{else}\ b_2 \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=}$$
$$(\mathrm{C}[\![\, b_1 \,]\!] \circ \mathrm{C}[\![\, e \bowtie 0 \,]\!])\mathcal{X} \cup (\mathrm{C}[\![\, b_2 \,]\!] \circ \mathrm{C}[\![\, e \not\bowtie 0 \,]\!])\mathcal{X}$$

$$\mathrm{C}[\![\, \mathtt{while}\ e \bowtie 0\ \mathtt{do}\ b\ \mathtt{done} \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=}$$
$$\mathrm{C}[\![\, e \not\bowtie 0 \,]\!]\,(\mathrm{lfp}\,\lambda\mathcal{Y}.\mathcal{X} \cup (\mathrm{C}[\![\, b \,]\!] \circ \mathrm{C}[\![\, e \bowtie 0 \,]\!])\mathcal{Y})$$

Abstract semantics

Given by the abstract domain:

- sound $C^\sharp[\![\, V := e \,]\!]$, $C^\sharp[\![\, e \bowtie 0 \,]\!]$, $\cup^\sharp$
- sound and terminating $\triangledown$

Derived analysis: ... to the abstract

$$C^\sharp[\![\, b_1; b_2 \,]\!]\,\mathcal{X}^\sharp \stackrel{\text{def}}{=} C^\sharp[\![\, b_2 \,]\!]\,(C^\sharp[\![\, b_1 \,]\!]\,\mathcal{X}^\sharp)$$

$$C^\sharp[\![\, \texttt{if } e \bowtie 0 \texttt{ then } b_1 \texttt{ else } b_2 \,]\!]\,\mathcal{X}^\sharp \stackrel{\text{def}}{=} (C^\sharp[\![\, b_1 \,]\!] \circ C^\sharp[\![\, e \bowtie 0 \,]\!])\mathcal{X}^\sharp \cup^\sharp (C^\sharp[\![\, b_2 \,]\!] \circ C^\sharp[\![\, e \not\bowtie 0 \,]\!])\mathcal{X}^\sharp$$

$$C^\sharp[\![\, \texttt{while } e \bowtie 0 \texttt{ do } b \texttt{ done} \,]\!]\,\mathcal{X}^\sharp \stackrel{\text{def}}{=} C^\sharp[\![\, e \not\bowtie 0 \,]\!]\,(\lim \lambda \mathcal{Y}^\sharp.\mathcal{Y}^\sharp \triangledown (\mathcal{X}^\sharp \cup^\sharp (C^\sharp[\![\, b \,]\!] \circ C^\sharp[\![\, e \bowtie 0 \,]\!])\mathcal{Y}^\sharp))$$

The derived analysis is sound and terminates.

Interval domain

Interval lattice

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

[Cousot 76]

Galois connection: $\mathcal{P}(\mathbb{Q}) \underset{\alpha_b}{\overset{\gamma_b}{\leftrightarrows}} \mathcal{B}^\sharp \cup \{\, \bot^\sharp \,\}$

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

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

Partial order:

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

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

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

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

Interval abstract arithmetic operators

Based on interval arithmetic [Moore 66]

$$
\begin{array}{rcl}
[c, c']^\sharp & \stackrel{\text{def}}{=} & [c, c'] \\
-^\sharp [a, b] & \stackrel{\text{def}}{=} & [-b, -a] \\
[a, b] +^\sharp [c, d] & \stackrel{\text{def}}{=} & [a + c, b + d] \\
[a, b] -^\sharp [c, d] & \stackrel{\text{def}}{=} & [a - d, b - c] \\
[a, b] \times^\sharp [c, d] & \stackrel{\text{def}}{=} & [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \\
[a, b] /^\sharp [c, d] & \stackrel{\text{def}}{=} & \cdots
\end{array}
$$

where $\pm\infty \times 0 = 0$.

Interval abstract assignment

Abstract evaluation of expressions: $\mathrm{E}^\sharp[\![\, e \,]\!] : \mathcal{D}^\sharp \to \mathcal{B}^\sharp$

$$
\begin{array}{lcl}
\mathrm{E}^\sharp[\![\, [c,c'] \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & [c,c']^\sharp \\
\mathrm{E}^\sharp[\![\, \mathtt{V} \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & \mathcal{X}^\sharp(\mathtt{V}) \\
\mathrm{E}^\sharp[\![\, -e \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & -^\sharp\ \mathrm{E}^\sharp[\![\, e \,]\!]\, \mathcal{X}^\sharp \\
\mathrm{E}^\sharp[\![\, e_1 \diamond e_2 \,]\!]\, \mathcal{X}^\sharp & \stackrel{\text{def}}{=} & \mathrm{E}^\sharp[\![\, e_1 \,]\!]\, \mathcal{X}^\sharp \diamond^\sharp \mathrm{E}^\sharp[\![\, e_2 \,]\!]\, \mathcal{X}^\sharp
\end{array}
$$

Abstract assignment:

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

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

Interval abstract tests

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

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

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

<u>General case:</u> constraint programming (HC4)

Interval widening

Interval widening example:

$$\bot^\sharp \ \triangledown \ X^\sharp \ \stackrel{\text{def}}{=} \ X^\sharp$$

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

Unstable bounds are set to $\pm\infty$

Smarter widenings also exist...

Analysis with widening example

```
X:=0;
while • X<40 do
  X:=X+3
done
```

Analysis with widening example

```
X:=0;
while • X<40 do
  X:=X+3
done
```

We must compute:

$$C^\sharp[\![\, \mathtt{X} \geq 40 \,]\!]\, (\lim \lambda \mathcal{Y}^\sharp . \mathcal{Y}^\sharp \,\triangledown\, (\mathcal{X}^\sharp \cup^\sharp C^\sharp[\![\, \mathtt{X} := \mathtt{X}+3 \,]\!]\, (C^\sharp[\![\, \mathtt{X} < 40 \,]\!]\, \mathcal{Y}^\sharp)))$$

- $\mathcal{Y}_0^\sharp = \mathcal{X}^\sharp = [0,0]$
- $\mathcal{Y}_1^\sharp = \mathcal{Y}_0^\sharp \,\triangledown\, (\mathcal{X}^\sharp \cup^\sharp (\mathcal{Y}_0^\sharp +^\sharp [3,3])) = [0,0] \,\triangledown\, ([0,0] \cup^\sharp [3,3]) = [0,+\infty]$
- $\mathcal{Y}_2^\sharp = \mathcal{Y}_1^\sharp \,\triangledown\, (\mathcal{X}^\sharp \cup^\sharp (\mathcal{Y}_1^\sharp +^\sharp [3,3])) = [0,+\infty] \,\triangledown\, ([0,0] \cup^\sharp [3,42]) = \mathcal{Y}_1^\sharp$
- $C^\sharp[\![\, \mathtt{X} \geq 40 \,]\!]\, (\mathcal{Y}_2^\sharp) = [42,+\infty]$

Decreasing iterations: to improve the precision

- after stabilization, continue iterating without $\triangledown$ (use $\cap$)
- in our case, $\mathcal{Y}_3^\sharp = [0,42]$, so $C^\sharp[\![\, \mathtt{X} \geq 40 \,]\!]\, (\mathcal{Y}_3^\sharp) = [40,42]$

Polyhedra Domain

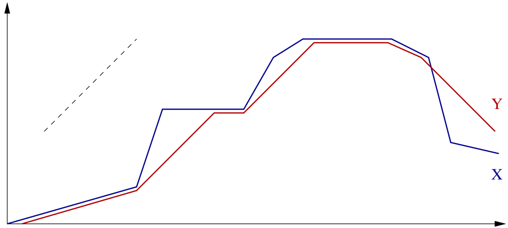
The need for relational domains

Non-relation domains cannot represent variable relationships

Rate limiter

```
Y:=0; while • true do
  X:=[-128,128]; D:=[0,16];
  S:=Y; Y:=X; R:=X-S;
  if R<=-D then Y:=S-D fi;
  if R>=D then Y:=S+D fi
done
```

- X: input signal
- Y: output signal
- S: last output
- R: delta Y-S
- D: max. allowed for |R|



The need for relational domains

Non-relation domains cannot represent variable relationships

Rate limiter

```
Y:=0; while • true do
  X:=[-128,128]; D:=[0,16];
  S:=Y; Y:=X; R:=X-S;
  if R<=-D then Y:=S-D fi;
  if R>=D then Y:=S+D fi
done
```

X: input signal
Y: output signal
S: last output
R: delta Y-S
D: max. allowed for |R|

Iterations in the interval domain (without widening):

$\mathcal{X}^{\sharp 0}_{\bullet}$	$\mathcal{X}^{\sharp 1}_{\bullet}$	$\mathcal{X}^{\sharp 2}_{\bullet}$	...	$\mathcal{X}^{\sharp n}_{\bullet}$
$Y = 0$	$\lvert Y \rvert \leq 144$	$\lvert Y \rvert \leq 160$	...	$\lvert Y \rvert \leq 128 + 16n$

In fact, $Y \in [-128, 128]$ always holds.

To prove that, e.g. $Y \geq -128$, we must be able to:

- represent the properties $R = X - S$ and $R \leq -D$
- combine them to deduce $S - X \geq D$, and then $Y = S - D \geq X$

Polyhedra domain

Domain proposed by [Cousot Halbwachs 78]
to infer conjunctions of affine inequalities $\bigwedge_j \left(\sum_{i=1}^{n} \alpha_{ij} \mathtt{V}_i \geq \beta_j\right)$.

Abstract elements:

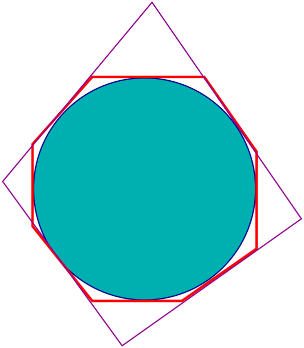
- *LinCons* $\stackrel{\text{def}}{=}$ affine constraints over $\mathcal{V}$ with coefficients in $\mathbb{Q}$
- $\mathcal{D}^\sharp \stackrel{\text{def}}{=} \mathcal{P}_{\text{finite}}(\mathit{LinCons})$

Concretization:

$$\gamma(\mathcal{X}^\sharp) \stackrel{\text{def}}{=} \{\, \rho \in \mathcal{V} \to \mathbb{Q} \mid \forall c \in \mathcal{X}^\sharp,\ \rho \models c \,\}$$

- $\gamma(\mathcal{X}^\sharp)$ is a closed convex polyhedron of $(\mathcal{V} \to \mathbb{Q}) \simeq \mathbb{Q}^{|\mathcal{V}|}$
- $\gamma(\mathcal{X}^\sharp)$ may be empty, bounded, or unbounded

Polyhedra representations



- No memory bound on the representations (even minimal ones)
- No best abstraction α
- Dual representation using generators
  (double description method)

Polyhedra algorithms

Fourier-Motzkin elimination:

Fourier$(\mathcal{X}^\sharp, \mathtt{V}_k)$ eliminates $\mathtt{V}_k$ from all the constraints in $\mathcal{X}^\sharp$:

$$\begin{array}{l} \textit{Fourier}(\mathcal{X}^\sharp, \mathtt{V}_k) \stackrel{\text{def}}{=} \\ \quad \{\ (\sum_i \alpha_i \mathtt{V}_i \geq \beta) \in \mathcal{X}^\sharp \mid \alpha_k = 0\ \} \cup \\ \quad \{\ (-\alpha_k^-)c^+ + \alpha_k^+ c^- \mid c^+ = (\sum_i \alpha_i^+ \mathtt{V}_i \geq \beta^+) \in \mathcal{X}^\sharp,\ \alpha_k^+ > 0, \\ \qquad\qquad\qquad\qquad\qquad\quad c^- = (\sum_i \alpha_i^- \mathtt{V}_i \geq \beta^-) \in \mathcal{X}^\sharp,\ \alpha_k^- < 0\ \} \end{array}$$

Semantics

$$\gamma(\textit{Fourier}(\mathcal{X}^\sharp, \mathtt{V}_k)) = \{\ \rho[\mathtt{V}_k \mapsto v] \mid v \in \mathbb{Q},\ \rho \in \gamma(\mathcal{X}^\sharp)\ \}$$

i.e., forget the value of $\mathtt{V}_k$

Polyhedra algorithms

Linear programming:

$$\textit{simplex}(\mathcal{X}^\sharp, \vec{\alpha}) \stackrel{\text{def}}{=} \min \{ \textstyle\sum_i \alpha_i \rho(\mathtt{V}_i) \mid \rho \in \gamma(\mathcal{X}^\sharp) \}$$

Application: remove redundant constraints:

for each $c = (\sum_i \alpha_i \mathtt{V}_i \geq \beta) \in \mathcal{X}^\sharp$
if $\beta \leq \textit{simplex}(\mathcal{X}^\sharp \setminus \{c\}, \vec{\alpha})$, then remove c from $\mathcal{X}^\sharp$

(e.g., *Fourier* causes a quadratic growth in constraint number, most of which are redundant)

Note: calling *simplex* many times can be costly

- use fast syntactic checks first
- check against the bounding-box first
- use *simplex* as a last resort

Polyhedra abstract operators

Order: $\subseteq^\sharp$

$$\mathcal{X}^\sharp \subseteq^\sharp \mathcal{Y}^\sharp \stackrel{\text{def}}{\iff} \forall (\textstyle\sum_i \alpha_i \mathtt{V}_i \geq \beta) \in \mathcal{Y}^\sharp,\ \textit{simplex}(\mathcal{X}^\sharp, \vec{\alpha}) \geq \beta$$

$$\stackrel{\text{def}}{\iff} \gamma(\mathcal{X}^\sharp) \subseteq \gamma(\mathcal{Y}^\sharp)$$

$$\mathcal{X}^\sharp =^\sharp \mathcal{Y}^\sharp \stackrel{\text{def}}{\iff} \mathcal{X}^\sharp \subseteq^\sharp \mathcal{Y}^\sharp \wedge \mathcal{Y}^\sharp \subseteq^\sharp \mathcal{X}^\sharp$$

Polyhedra abstract operators (cont.)

Convex hull:

- Express a point $\vec{V} \in \mathcal{X}^\sharp \cup^\sharp \mathcal{Y}^\sharp$ as a convex combination:
  $\vec{V} = \sigma\vec{X} + \sigma'\vec{Y}$ for $\vec{X} \in \mathcal{X}^\sharp$, $\vec{Y} \in \mathcal{Y}^\sharp$, $\sigma + \sigma' = 1$, $\sigma, \sigma' \geq 0$
- as $\sigma\vec{X} + \sigma'\vec{Y}$ is quadratic
  we consider instead: $\vec{V} = \vec{X} + \vec{Y}$ with $\vec{X}/\sigma \in \mathcal{X}^\sharp$, $\vec{Y}/\sigma' \in \mathcal{Y}^\sharp$
  i.e., $\vec{X} \in \sigma\mathcal{X}^\sharp$, $\vec{Y} \in \sigma'\mathcal{Y}^\sharp$
  (adds closure points on unbounded polyhedra)

Formally:

$$\mathcal{X}^\sharp \cup^\sharp \mathcal{Y}^\sharp \stackrel{\text{def}}{=}$$

$$\begin{array}{ll} \textit{Fourier}(& \{ (\sum_j \alpha_j X_j - \beta\sigma \geq 0) \mid (\sum_j \alpha_j V_j \geq \beta) \in \mathcal{X}^\sharp \} \ \cup \\ & \{ (\sum_j \alpha_j Y_j - \beta\sigma' \geq 0) \mid (\sum_j \alpha_j V_j \geq \beta) \in \mathcal{Y}^\sharp \} \ \cup \\ & \{ V_j = X_j + Y_j \mid V_j \in \mathcal{V} \} \cup \{ \sigma \geq 0,\ \sigma' \geq 0,\ \sigma + \sigma' = 1 \}, \\ & \{ X_j, Y_j \mid V_j \in \mathcal{V} \} \ \cup \ \{ \sigma, \sigma' \}) \end{array}$$

[Benoi et al. 96]

Polyhedra abstract operators (cont.)

Precise abstract commands: (exact)

$$\mathrm{C}^\sharp[\![\, \sum_i \alpha_i \mathtt{V}_i + \beta \le \mathtt{0} \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \mathcal{X}^\sharp \cup \{(\sum_i \alpha_i \mathtt{V}_i + \beta \le \mathtt{0})\}$$

$$\mathrm{C}^\sharp[\![\, \mathtt{V}_j := [-\infty, +\infty] \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=} \mathit{Fourier}(\mathcal{X}^\sharp, \mathtt{V}_j))$$

$$\mathrm{C}^\sharp[\![\, \mathtt{V}_j := \sum_i \alpha_i \mathtt{V}_i + \beta^\sharp \,]\!]\, \mathcal{X}^\sharp \stackrel{\text{def}}{=}$$
$$\mathit{subst}(\mathtt{V} \mapsto \mathtt{V}_i,\ \mathit{Fourier}((\mathcal{X}^\sharp \cup \{\mathtt{V} = \sum_i \alpha_i \mathtt{V}_i + \beta\}),\ \mathtt{V}_j))$$

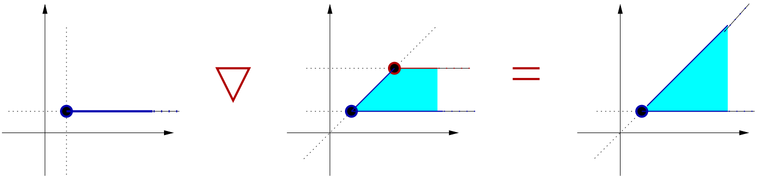
Polyhedra widening

Classic widening $\triangledown$ in $\mathcal{D}^\sharp$

$$\mathcal{X}^\sharp \triangledown \mathcal{Y}^\sharp \stackrel{\text{def}}{=} \{\, c \in \mathcal{X}^\sharp \mid \mathcal{Y}^\sharp \subseteq^\sharp \{c\} \,\}$$

- suppress unstable constraints

Example:



Floating-point domains

Floating-point uses

Two independent problems:

- **Implement the analyzer using floating-point**

  goal: trade precision for efficiency

  exact rational arithmetic can be costly
  coefficients can grow large (polyhedra)

- **Analyze floating-point programs**

  goal: catch run-time errors caused by rounding
  (overflow, division by 0, . . .)

Also: a floating-point analyzer for floating-point programs.

Challenge: how to stay **sound**?

Floating-point expressions

Floating-point expressions $\texttt{exp}_f$

$\texttt{exp}_f$	$::=$	$[c, c']$	constant range $c, c' \in \mathbb{F}$, $c \leq c'$
	$\mid$	$\texttt{V}$	variable $\texttt{V} \in \mathcal{V}$
	$\mid$	$\ominus\ \texttt{exp}_f$	negation
	$\mid$	$\texttt{exp}_f \odot_r \texttt{exp}_f$	operator $\odot \in \{\oplus, \ominus, \otimes, \oslash\}$

(we use circled operators to distinguish them from operators in $\mathbb{Q}$)

Concrete semantics of expressions

Semantics of rounding: $R_r \colon \mathbb{Q} \to \mathbb{F} \cup \{\mathcal{O}\}$.

Example definition:

$$R_{+\infty}(x) \stackrel{\text{def}}{=} \begin{cases} \min \{\, y \in \mathbb{F} \mid y \geq x \,\} & \text{if } x \leq Mf \\ \mathcal{O} & \text{if } x > Mf \end{cases}$$

$$R_{-\infty}(x) \stackrel{\text{def}}{=} \begin{cases} \max \{\, y \in \mathbb{F} \mid y \leq x \,\} & \text{if } x \geq -Mf \\ \mathcal{O} & \text{if } x < -Mf \end{cases}$$

Notes:

- $\forall x, r,\ R_{-\infty}(x) \leq R_r(x) \leq R_{+\infty}(x)$
- $\forall r, R_r$ is monotonic

Concrete semantics of expressions (cont.)

$\mathrm{E}[\![\, e_f \,]\!] : (\mathcal{V} \to \mathbb{F}) \to \mathcal{P}(\mathbb{F} \cup \{\mathcal{O}\})$ (expression semantics)

$$\mathrm{E}[\![\, \mathtt{V} \,]\!]\, \rho \stackrel{\text{def}}{=} \{\, \rho(\mathtt{V}) \,\}$$

$$\mathrm{E}[\![\, [c, c'] \,]\!]\, \rho \stackrel{\text{def}}{=} \{\, x \in \mathbb{F} \mid c \leq x \leq c' \,\}$$

$$\mathrm{E}[\![\, \ominus\, e_f \,]\!]\, \rho \stackrel{\text{def}}{=} \{\, -x \mid x \in \mathrm{E}[\![\, e_f \,]\!]\, \rho \cap \mathbb{F} \,\} \cup (\{\mathcal{O}\} \cap \mathrm{E}[\![\, e_f \,]\!]\, \rho)$$

$$\begin{aligned}\mathrm{E}[\![\, e_f \odot_r e'_f \,]\!]\, \rho \stackrel{\text{def}}{=}\; & \{\, R_r(x \cdot y) \mid x \in \mathrm{E}[\![\, e_f \,]\!]\, \rho \cap \mathbb{F},\ y \in \mathrm{E}[\![\, e'_f \,]\!]\, \rho \cap \mathbb{F} \,\} \cup \\ & \{\, \mathcal{O} \mid \text{ if } \mathcal{O} \in \mathrm{E}[\![\, e_f \,]\!]\, \rho \cup \mathrm{E}[\![\, e'_f \,]\!]\, \rho \,\} \\ & \{\, \mathcal{O} \mid \text{ if } 0 \in \mathrm{E}[\![\, e'_f \,]\!]\, \rho \text{ and } \odot = \oslash \,\}\end{aligned}$$

$\mathrm{C}[\![\, c \,]\!] : \mathcal{P}(\mathcal{V} \to \mathbb{F}) \to \mathcal{P}((\mathcal{V} \to \mathbb{F}) \cup \{\mathcal{O}\})$ (command semantics)

$$\begin{aligned}\mathrm{C}[\![\, \mathtt{X} := e_f \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=}\; & \{\, \rho[\, \mathtt{X} \mapsto v \,] \mid \rho \in \mathcal{X},\ v \in \mathrm{E}[\![\, e_f \,]\!]\, \rho \cap \mathbb{F} \,\} \\ & \cup (\{\mathcal{O}\} \cap \mathrm{E}[\![\, e_f \,]\!]\, \mathcal{X})\end{aligned}$$

$$\begin{aligned}\mathrm{C}[\![\, e_f \leq 0 \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=}\; & \{\, \rho \mid \rho \in \mathcal{X},\ \exists v \in \mathrm{E}[\![\, e_f \,]\!]\, \rho \cap \mathbb{F},\ v \leq 0 \,\} \\ & \cup (\{\mathcal{O}\} \cap \mathrm{E}[\![\, e_f \,]\!]\, \mathcal{X})\end{aligned}$$

Floating-point interval domain

Representation: $\mathcal{B}^\sharp \stackrel{\text{def}}{=} \{\, [a,b] \mid a \in \mathbb{F},\ b \in \mathbb{F},\ a \le b \,\}$

Expression semantics: $\mathrm{E}^\sharp[\![\, \mathtt{exp}_f \,]\!] : (\mathcal{V} \to \mathcal{B}^\sharp) \to (\mathcal{B}^\sharp \cup \{\, \mathcal{O} \,\})$

$$
\begin{array}{lcl}
[a,b] \oplus^\sharp [a',b'] & \stackrel{\text{def}}{=} & [R_{-\infty}(a+a'), R_{+\infty}(b+b')] \\
[a,b] \ominus^\sharp [a',b'] & \stackrel{\text{def}}{=} & [R_{-\infty}(a-b'), R_{+\infty}(b-a')] \\
[a,b] \otimes^\sharp [a',b'] & \stackrel{\text{def}}{=} & [\ R_{-\infty}(\min(aa',ab',ba',bb')), \\
 & & R_{+\infty}(\max(aa',ab',ba',bb'))]
\end{array}
$$

- We suppose r is unknown and assume a worst case rounding.
- Soundness stems from the monotonicity of $R_{-\infty}$ and $R_{+\infty}$.
- Abstract operators also use float arithmetic (efficiency).

Expression linearization

Floating-point issues in relational domains

Relational domains assume many powerful properties on $\mathbb{Q}$: associativity, distributivity,... that are not true on $\mathbb{F}$!

Example: Fourier-Motzkin elimination

$$\texttt{X} - \texttt{Y} \leq c \quad \wedge \quad \texttt{Y} - \texttt{Z} \leq d \quad \Longrightarrow \quad \texttt{X} - \texttt{Z} \leq c + d$$

$$\texttt{X} \ominus_n \texttt{Y} \leq c \quad \wedge \quad \texttt{Y} \ominus_n \texttt{Z} \leq d \quad \not\Longrightarrow \quad \texttt{X} \ominus_n \texttt{Z} \leq c \oplus_n d$$

$$(\texttt{X} = 1,\ \texttt{Y} = 10^{38},\ \texttt{Z} = -1,\ c = \texttt{X} \ominus_n \texttt{Y} = -10^{38},$$
$$d = \texttt{Y} \ominus_n \texttt{Z} = 10^{38},\ c \oplus_n d = 0,\ \texttt{X} \ominus_n \texttt{Z} = 2 > 0)$$

We cannot manipulate float expressions as easily as rational ones!

Solution:

keep representing and manipulating rational expressions

- abstract float expressions from programs into rational ones
- feed them to a rational abstract domain
- (optional) implement the rational domain using floats

Affine interval forms

We put expressions in affine interval form: [Miné 04]

$$\texttt{exp}_\ell ::= [a_0, b_0] + \sum_k [a_k, b_k] \times \texttt{V}_k$$

Semantics:

$$\mathrm{E}[\![\, e_\ell \,]\!]\, \rho \stackrel{\text{def}}{=} \{\, c_0 + \sum_k c_k \times \rho(\texttt{V}_k) \mid \forall i,\ c_i \in [a_i, b_i] \,\}$$

(evaluated in $\mathbb{Q}$)

Advantages:

- affine expressions are easy to manipulate
- interval coefficients allow non-determinism in expressions, hence, the opportunity for abstraction
- intervals can easily model rounding errors
- easy to design algorithms for $\mathrm{C}^\sharp[\![\, \texttt{X} := e_\ell \,]\!]$ and $\mathrm{C}^\sharp[\![\, e_\ell \le 0 \,]\!]$ in most domains

Affine interval form algebra

Operations on affine interval forms:

- adding $\boxplus$ and subtracting $\boxminus$ two forms
- multiplying $\boxtimes$ and dividing $\boxslash$ a form by an interval

Using interval arithmetic $\oplus^\sharp$, $\ominus^\sharp$, $\otimes^\sharp$, $\oslash^\sharp$:

$$(i_0 + \sum_k i_k \times \mathtt{V}_k) \boxplus (i'_0 + \sum_k i'_k \times \mathtt{V}_k) \stackrel{\text{def}}{=} (i_0 \oplus^\sharp i'_0) + \sum_k (i_k \oplus^\sharp i'_k) \times \mathtt{V}_k$$

$$i \boxtimes (i_0 + \sum_k i_k \times \mathtt{V}_k) \stackrel{\text{def}}{=} (i \otimes^\sharp i_0) + \sum_k (i \otimes^\sharp i_k) \times \mathtt{V}_k$$

...

Intervalization: $\iota : (\mathtt{exp}_\ell \times \mathcal{D}^\sharp) \to \mathtt{exp}_\ell$

Intervalization flattens the expression into a single interval:

$$\iota(i_0 + \sum_k i_k \times \mathtt{V}_k,\ \mathcal{X}^\sharp) \stackrel{\text{def}}{=} i_0 \oplus^\sharp \sum\nolimits_k^\sharp (i_k \otimes^\sharp \pi_k(\mathcal{X}^\sharp)).$$

Linearization of rational expressions

Linearization without rounding errors: $\ell : (\texttt{exp} \times \mathcal{D}^\sharp) \to \texttt{exp}_\ell$

Defined by induction on the syntax of expressions:

- $\ell(\texttt{V}, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} [1,1] \times \texttt{V}$
- $\ell([a,b], \mathcal{X}^\sharp) \stackrel{\text{def}}{=} [a,b]$
- $\ell(e_1 + e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} \ell(e_1, \mathcal{X}^\sharp) \boxplus \ell(e_2, \mathcal{X}^\sharp)$
- $\ell(e_1 - e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} \ell(e_1, \mathcal{X}^\sharp) \boxminus \ell(e_2, \mathcal{X}^\sharp)$
- $\ell(e_1 / e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} \ell(e_1, \mathcal{X}^\sharp) \boxslash \iota(\ell(e_2, \mathcal{X}^\sharp), \mathcal{X}^\sharp)$
- $\ell(e_1 \times e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=}$ can be $\begin{cases} \text{either} & \iota(\ell(e_1, \mathcal{X}^\sharp), \mathcal{X}^\sharp) \boxtimes \ell(e_2, \mathcal{X}^\sharp) \\ \text{or} & \iota(\ell(e_2, \mathcal{X}^\sharp), \mathcal{X}^\sharp) \boxtimes \ell(e_1, \mathcal{X}^\sharp) \end{cases}$

Linearization of floating-point expressions

Rounding an affine interval form: (32-bit single precision)

- if the result is normalized: we have a relative error ε with magnitude 2^{-23}:

$$\varepsilon([a_0, b_0] + \sum_k [a_k, b_k] \times \mathtt{V}_k) \stackrel{\text{def}}{=} \max(|a_0|, |b_0|) \times [-2^{-23}, 2^{-23}] + \sum_k (\max(|a_k|, |b_k|) \times [-2^{-23}, 2^{-23}] \times \mathtt{V}_k)$$

- if the result is denormalized, we have an absolute error $\omega \stackrel{\text{def}}{=} [-2^{-149}, 2^{-149}]$.

$\Longrightarrow$ we sum these two sources of rounding errors

Linearization with rounding errors: $\ell : (\mathtt{exp}_f \times \mathcal{D}^\sharp) \to \mathtt{exp}_\ell$

$$\ell(e_1 \oplus_r e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} \ell(e_1, \mathcal{X}^\sharp) \boxplus \ell(e_2, \mathcal{X}^\sharp) \boxplus \varepsilon(\ell(e_1, \mathcal{X}^\sharp)) \boxplus \varepsilon(\ell(e_2, \mathcal{X}^\sharp)) \boxplus \omega$$

$$\ell(e_1 \otimes_r e_2, \mathcal{X}^\sharp) \stackrel{\text{def}}{=} \iota(\ell(e_1, \mathcal{X}^\sharp), \mathcal{X}^\sharp) \boxtimes (\ell(e_2, \mathcal{X}^\sharp) \boxplus \varepsilon(\ell(e_2, \mathcal{X}^\sharp))) \boxplus \omega$$

...

Applications of the floating-point linearization

Soundness of the linearization

$\forall e,\ \forall \mathcal{X}^\sharp \in \mathcal{D}^\sharp,\ \forall \rho \in \gamma(\mathcal{X}^\sharp),$
if $\mathcal{O} \notin \mathrm{E}[\![\, e \,]\!]\, \rho$, then $\mathrm{E}[\![\, e \,]\!]\, \rho \subseteq \mathrm{E}[\![\, \ell(e, \mathcal{X}^\sharp) \,]\!]\, \rho$

Application: $\mathrm{C}^\sharp[\![\, \mathtt{V} := e \,]\!]\, \mathcal{X}^\sharp$

- check that $\mathcal{O} \notin \mathrm{E}[\![\, e \,]\!]\, \rho$ for $\rho \in \gamma(\mathcal{X}^\sharp)$ with interval arithmetic
- compute $\mathrm{C}^\sharp[\![\, \mathtt{V} := e \,]\!]\, \mathcal{X}^\sharp$ as $\mathrm{C}^\sharp[\![\, \mathtt{V} := \ell(e, \mathcal{X}^\sharp) \,]\!]\, \mathcal{X}^\sharp$
- (use $\mathrm{C}^\sharp[\![\, \mathtt{V} := [-Mf, Mf] \,]\!]\, \mathcal{X}^\sharp$ if $\mathcal{O} \in \mathrm{E}[\![\, e \,]\!]\, \rho$)

Application : the Astrée analyzer

The Astrée static analyzer

Analyseur statique de programmes temps-réels embarqués

(static analyzer for real-time embedded software)

- developed at ENS (since 2001)

  B. Blanchet, P. Cousot, R. Cousot, J. Feret,
  L. Mauborgne, D. Monniaux, A. Miné, X. Rival

- industrialized and made commercially available by AbsInt (since 2009)

Astrée

`www.astree.ens.fr`

AbsInt

`www.absint.com`

[Blanchet et al. 03]

Specialized static analyzers

Design by refinement:

- focus on a specific family of programs and properties
- start with a fast and coarse analyzer (intervals)
- while the precision is insufficient (too many false alarms)
  - add new abstract domains (generic or application-specific)
  - refine existing domains (better transfer functions)
  - improve communication between domains (reductions)

$\Longrightarrow$ analyzer specialized for a (infinite) class of programs

- efficient and precise
- parametric (by end-users, to analyze new programs in the family)
- extensible (by developers, to analyze related families)

Astrée applications

Airbus A340-300 (2003)

Airbus A380 (2004)

(model of) ESA ATV (2008)

- size: from 70 000 to 860 000 lines of C
- analysis time: from 45mn to $\simeq$40h
- alarm(s): 0 (proof of absence of run-time error)

Sound floating-point polyhedra

Sound floating-point polyhedra

Algorithms to adapt: [Chen al. 08]

- linear programming:

$$simplex_f(\mathcal{X}^\sharp, \vec{\alpha}) \leq simplex(\mathcal{X}^\sharp, \vec{\alpha})$$

$$simplex(\mathcal{X}^\sharp, \vec{\alpha}) \stackrel{\text{def}}{=} \min \{ \textstyle\sum_k \alpha_k \rho(\mathtt{V}_k) \mid \rho \in \gamma(\mathcal{X}^\sharp) \}$$

- Fourier-Motzkin elimination:

$$Fourier_f(\mathcal{X}^\sharp, \mathtt{V}_k) \Longleftarrow Fourier(\mathcal{X}^\sharp, \mathtt{V}_k)$$

$$\begin{aligned} Fourier(\mathcal{X}^\sharp, \mathtt{V}_k) \stackrel{\text{def}}{=} & \\ & \{ (\textstyle\sum_i \alpha_i \mathtt{V}_i \geq \beta) \in \mathcal{X}^\sharp \mid \alpha_k = 0 \} \cup \\ & \{ (-\alpha_k^-)c^+ + \alpha_k^+ c^- \mid c^+ = (\textstyle\sum_i \alpha_i^+ \mathtt{V}_i \geq \beta^+) \in \mathcal{X}^\sharp,\ \alpha_k^+ > 0, \\ & \qquad c^- = (\textstyle\sum_i \alpha_i^- \mathtt{V}_i \geq \beta^-) \in \mathcal{X}^\sharp,\ \alpha_k^- < 0 \} \end{aligned}$$

Sound floating-point linear programming

Guaranteed linear programming: [Neumaier Shcherbina 04]

Goal: under-approximate $\mu = \min \{ \vec{c} \cdot \vec{x} \mid \mathbf{M} \times \vec{x} \leq \vec{b} \}$

knowing that $\vec{x} \in [\vec{x}_l, \vec{x}_h]$ (bounding-box for $\gamma(\mathcal{X}^\sharp)$).

- compute any approximation $\tilde{\mu}$ of the dual problem:
  $\tilde{\mu} \simeq \mu = \max \{ \vec{b} \cdot \vec{y} \mid {}^t\mathbf{M} \times \vec{y} = \vec{c},\ \vec{y} \leq \vec{0} \}$
  and the corresponding vector $\vec{y}$

  (e.g. using an off-the-shelf solver; $\tilde{\mu}$ may over-approximate or under-approximate μ)

- compute with intervals safe bounds $[\vec{r}_l, \vec{r}_h]$ for $\mathbf{A} \times \vec{y} - \vec{c}$:
  $[\vec{r}_l, \vec{r}_h] = ({}^t\mathbf{A} \otimes^\sharp \vec{y}) \ominus^\sharp \vec{c}$

  and then:
  $\nu = \inf((\vec{b} \otimes^\sharp \vec{y}) \ominus^\sharp ([\vec{r}_l, \vec{r}_h] \otimes^\sharp [\vec{x}_l, \vec{x}_h]))$

then: $\nu \leq \mu$.

Sound floating-point Fourier-Motzkin elimination

Given:

- $c^+ = (\sum_i \alpha_i^+ \mathtt{V}_i \geq \beta^+)$ with $\alpha_k^+ > 0$
- $c^- = (\sum_i \alpha_i^- \mathtt{V}_i \geq \beta^-)$ with $\alpha_k^- < 0$
- a bounding-box of $\gamma(\mathcal{X}^\sharp)$: $[\vec{x}_l, \vec{x}_h]$

We wish to compute $\sum_{i \neq k} \alpha_i \mathtt{V}_i \geq \beta$ in $\mathbb{F}$
implied by $(-\alpha_k^-) c^+ + \alpha_k^+ c^-$ in $\gamma(\mathcal{X}^\sharp)$.

- normalize c^+ and c^- using interval arithmetic:

  $$\begin{cases} \mathtt{V}_k + \sum_{i \neq k} (\alpha_i^+ \oslash^\sharp \alpha_k^+) \mathtt{V}_i \geq \beta^+ \oslash^\sharp \alpha_k^+ \\ -\mathtt{V}_k + \sum_{i \neq k} (\alpha_i^- \oslash^\sharp (-\alpha_k^-)) \mathtt{V}_i \geq \beta^- \oslash^\sharp (-\alpha_k^-) \end{cases}$$

  (interval affine forms)

- add them using interval arithmetic:

  $$\sum_{i \neq k} [a_i, b_i] \mathtt{V}_i \geq [a_0, b_0]$$

  where $[a_i, b_i] = (\alpha_i^+ \oslash^\sharp \alpha_k^+) \ominus^\sharp (\alpha_i^- \oslash^\sharp \alpha_k^-)$,
  $[a_0, b_0] = (\beta^+ \oslash^\sharp \alpha_k^+) \ominus^\sharp (\beta^- \oslash^\sharp \alpha_k^-)$.

Sound floating-point Fourier-Motzkin elimination (cont.)

- linearize the interval affine form $\sum_{i \neq k} [a_i, b_i] \mathtt{V}_i \geq [a_0, b_0]$ into an affine form $\sum_{i \neq k} \alpha_i \mathtt{V}_i \geq \beta$

  we choose:

  - $\alpha_i \in [a_i, b_i]$
  - $\beta = \sup \left([a_0, b_0] \oplus^\sharp \bigoplus^\sharp_{i \neq k} (|\alpha_i \ominus^\sharp [a_i, b_i]|) \otimes^\sharp |[\vec{x}_l, \vec{x}_h]|\right)$

Soundness:

For all choices of $\alpha_i \in [a_i, b_i]$,
$\sum_{i \neq k} \alpha_i \mathtt{V}_k \geq \beta$ holds in $\mathit{Fourier}(\mathcal{X}^\sharp, \mathtt{V}_k)$.
(e.g. $\alpha_i = (a_i \oplus_n b_i) \oslash 2$)

Consequences of rounding

Precision loss:

- Projection:

$$\gamma(\mathit{Fourier}_f(\mathcal{X}^\sharp, \mathtt{V}_k)) \supseteq \{\, \rho[\mathtt{V}_k \mapsto v] \mid v \in \mathbb{Q},\ \rho \in \gamma(\mathcal{X}^\sharp) \,\} = \mathrm{C}[\![\, \mathtt{V}_k := [-\infty, +\infty] \,]\!]\, \gamma(\mathcal{X}^\sharp)$$

- Order:

$$\mathcal{X}^\sharp \subseteq^\sharp \mathcal{Y}^\sharp \Longrightarrow \gamma(\mathcal{X}^\sharp) \subseteq \gamma(\mathcal{Y}^\sharp) \quad (\not\Leftarrow)$$

- Join:

$$\gamma(\mathcal{X}^\sharp \cup^\sharp \mathcal{Y}^\sharp) \supseteq \mathit{ConvexHull}(\gamma(\mathcal{X}^\sharp) \cup \gamma(\mathcal{Y}^\sharp)) \quad (\neq)$$

Efficiency loss:

- cannot remove all redundant constraints

Abstraction summary

Floating-point polyhedra analyzer for floating-point programs

expression abstraction

float expression e_f

$\downarrow$ linearization

affine form e_ℓ in $\mathbb{Q}$

$\downarrow$ float implementation

affine form e_ℓ in $\mathbb{F}$ $\longrightarrow$

environment abstraction

$\mathcal{P}(\mathcal{V} \to \mathbb{F})$

$\downarrow$ abstract domain

polyhedra in $\mathbb{Q}$

$\downarrow$ float implementation

polyhedra in $\mathbb{F}$

$\downarrow$ widening

polyhedra in $\mathbb{F}$

The end