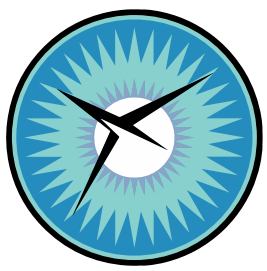
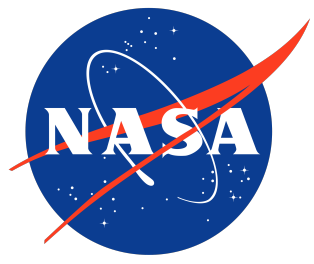
PRECiSA: a static analysis tool for floating-point programs

Laura Titolo

joint work with
Mariano Moscato, Marco A. Feliu, César A. Muñoz,
Paolo Masci, and Aaron Dutle

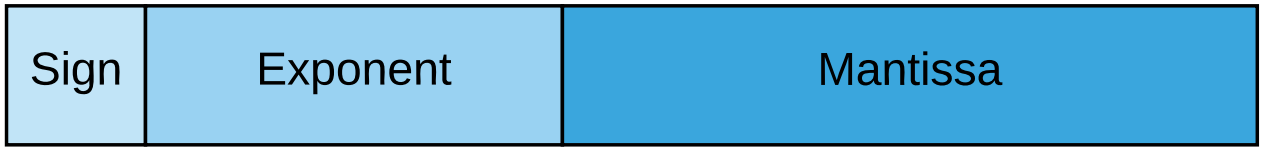




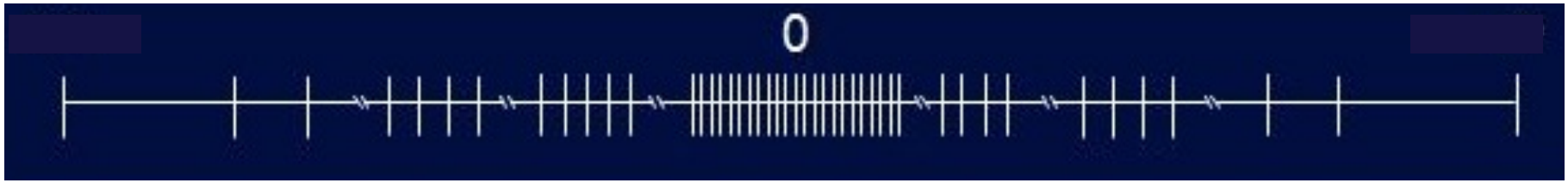
NIA/NASA LaRC

Motivations

- Floating-point numbers $\mathbb{F}$ = finite representation of Reals $\mathbb{R}$



- Not every real number can be represented as a floating-point
- ulp = unit in the last place
  = difference between two consecutive FP numbers



Motivations

- Round-off errors = difference between $\mathbb{R}$ and its $\mathbb{F}$ representation
  $\Rightarrow$ computed $\mathbb{F}$ result $\neq$ expected $\mathbb{R}$ result
- Unstable guards = $\mathbb{F}$ control-flow $\neq$ $\mathbb{R}$ control-flow
- Possible consequences in safety-critical aviation software:
  - ADS-B Compact Position Reporting (CPR) $\Rightarrow$ position errors as large as 360 nmi
  - Air traffic conflict and collision avoidance systems $\Rightarrow$ resolution maneuvers that are not implicitly coordinated
  - Geofencing in autonomous UAS $\Rightarrow$ incorrect determination of being inside/outside a geofence, even when the aircraft is far from the boundary

PRECiSA

PRECiSA is a framework for the analysis of floating-point programs

- computes round-off error bounds
  + formal proof of their correctness
- generates stable C code from a real specification instrumented to detect unstable guards

PRECiSA is a framework for the analysis of floating-point programs

- **computes round-off error bounds**
  + formal proof of their correctness
- generates stable C code from a real specification instrumented to detect unstable guards

PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

Output: for each $\tilde{f}$ find a round-off error bound e such that

$$|\tilde{f} - f| \leq e$$

PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

Output: for each $\tilde{f}$ find a round-off error bound e such that

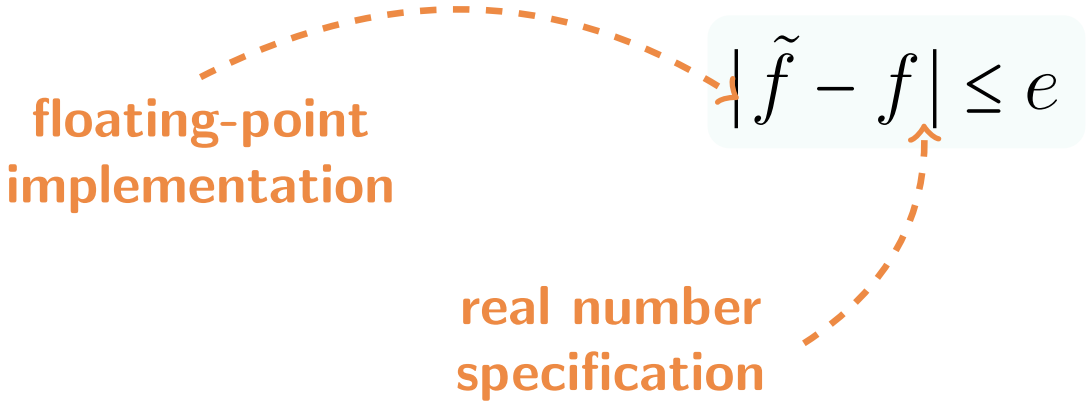
$$|\tilde{f} - f| \leq e$$

floating-point implementation

PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

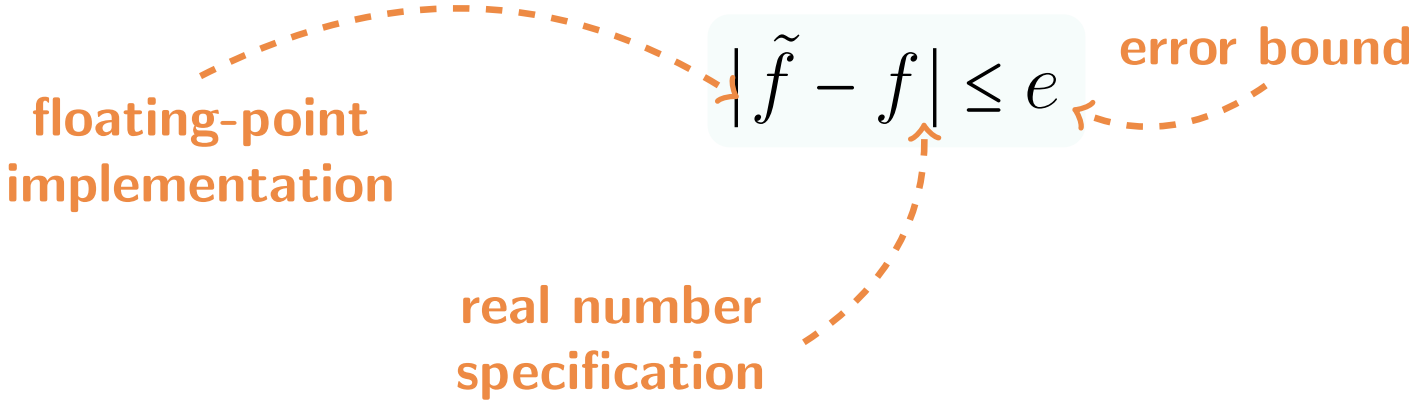
Output: for each $\tilde{f}$ find a round-off error bound e such that



PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

Output: for each $\tilde{f}$ find a round-off error bound e such that



PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

Output: for each $\tilde{f}$ find a round-off error bound e such that

$$|\tilde{f} - f| \leq e$$

Challenges:

- e should be tight (no coarse over-estimations)
- correctly bound the rounding errors in conditionals

PRECiSA Analysis

Input: program $\widetilde{P}$ = set of floating-point function declarations $\tilde{f}$

Output: for each $\tilde{f}$ find a round-off error bound e such that

$$|\tilde{f} - f| \leq e$$

Challenges:

- e should be tight (no coarse over-estimations)
- correctly bound the rounding errors in conditionals

Example

$$sign(\tilde{x}) = if\ (\tilde{x} \geq 0)\ then\ 1\ else\ -1$$

If $\tilde{x} \geq 0$ is true in $\mathbb{F}$ but false in $\mathbb{R}$, the error is 2.

Formalization of Floating-Point Round-off Errors

Round-off Error: $|\mathbf{F}(r) - r| \leq \frac{1}{2}\mathsf{ulp}(r)$

Formalization of Floating-Point Round-off Errors

Round-off Error: $|\mathbf{F}(r) - r| \leq \frac{1}{2}\mathsf{ulp}(r)$

unit in the last place

rounding of r

Formalization of Floating-Point Round-off Errors

Round-off Error: $|\mathbf{F}(r) - r| \leq \frac{1}{2}\mathsf{ulp}(r)$

$$|\widetilde{\mathsf{op}}(\tilde{v}_1 \ldots \tilde{v}_n) - \mathsf{op}(r_1 \ldots r_n)| \leq \epsilon_{\mathsf{op}}(e_1 \ldots e_n) + \tfrac{1}{2}\,\mathsf{ulp}(\upsilon(r_1 \ldots r_n, e_1 \ldots e_n))$$

error propagation (operands)

round-off error of the result

Formalization of Floating-Point Round-off Errors

Round-off Error: $|\mathbf{F}(r) - r| \leq \frac{1}{2}\mathsf{ulp}(r)$

$$|\widetilde{\mathsf{op}}(\tilde{v}_1 \ldots \tilde{v}_n) - \mathsf{op}(r_1 \ldots r_n)| \leq \epsilon_{\mathsf{op}}(e_1 \ldots e_n) + \tfrac{1}{2}\,\mathsf{ulp}(\upsilon(r_1 \ldots r_n, e_1 \ldots e_n))$$

Examples

$$|\,(r_1 + r_2) - (\tilde{v}_1 \tilde{+} \tilde{v}_2)\,| \leq e_1 + e_2 + 1/2\,\mathsf{ulp}(|r_1 + r_2| + e_1 + e_2)$$

$$|\,(r_1 * r_2) - (\tilde{v}_1 \tilde{*} \tilde{v}_2)\,| \leq |r_1|e_2 + |r_2|e_1 + e_1 e_2 + 1/2\,\mathsf{ulp}((|r_1| + e_1)(|r_2| + e_2))$$

Abstract Domain: Conditional Error Bounds

For each function declaration $\tilde{f}$

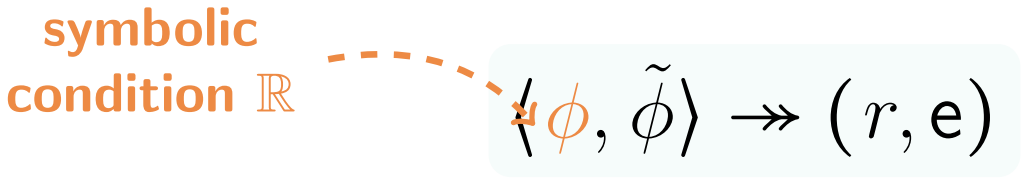
PRECiSA computes a set of **conditional error bounds**

$$\langle \phi, \tilde{\phi} \rangle \twoheadrightarrow (r, \mathsf{e})$$

Abstract Domain: Conditional Error Bounds

For each function declaration $\tilde{f}$

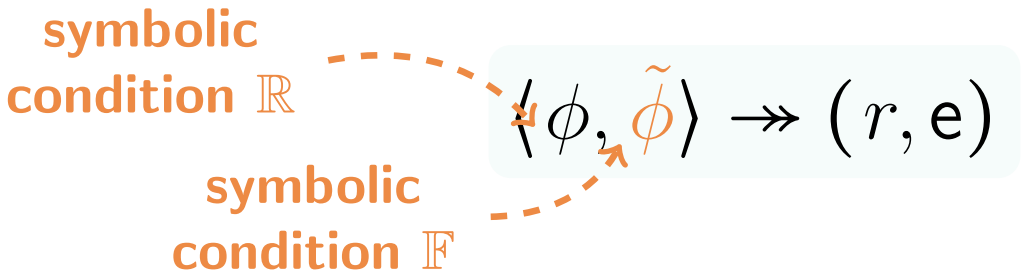
PRECiSA computes a set of **conditional error bounds**



Abstract Domain: Conditional Error Bounds

For each function declaration $\tilde{f}$

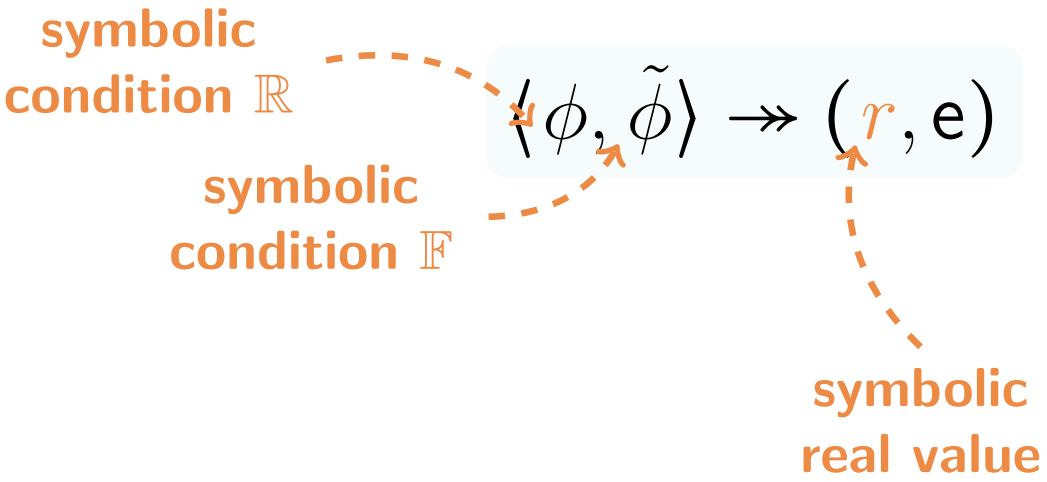
PRECiSA computes a set of **conditional error bounds**



Abstract Domain: Conditional Error Bounds

For each function declaration $\tilde{f}$

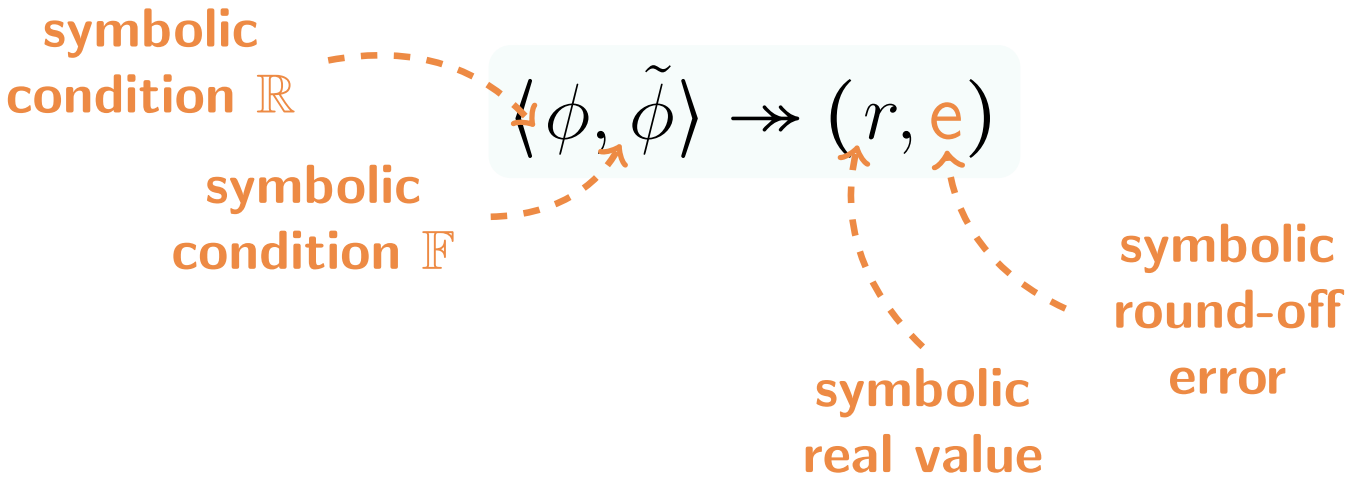
PRECiSA computes a set of **conditional error bounds**



Abstract Domain: Conditional Error Bounds

For each function declaration $\tilde{f}$

PRECiSA computes a set of **conditional error bounds**



Semantics of mathematical operators

From the semantics of the operands...

$\langle \phi_1, \tilde{\phi}_1 \rangle \twoheadrightarrow (r_1, e_1) \in \mathcal{E}[\![\tilde{a}_1]\!]_{(\sigma,I)}$ $\quad$ $\langle \phi_2, \tilde{\phi}_2 \rangle \twoheadrightarrow (r_2, e_2) \in \mathcal{E}[\![\tilde{a}_2]\!]_{(\sigma,I)}$

Semantics of mathematical operators

From the semantics of the operands...

$$\langle \phi_1, \tilde{\phi}_1 \rangle \twoheadrightarrow (r_1, e_1) \in \mathcal{E}[\![\tilde{a}_1]\!]_{(\sigma,I)} \qquad \langle \phi_2, \tilde{\phi}_2 \rangle \twoheadrightarrow (r_2, e_2) \in \mathcal{E}[\![\tilde{a}_2]\!]_{(\sigma,I)}$$

...we build the semantics of the operator bottom-up

Division $\tilde{a}_1 \tilde{/} \tilde{a}_2$

$$\langle \phi_1 \wedge \phi_2 \wedge r_2 \neq 0, \tilde{\phi}_1 \wedge \tilde{\phi}_2 \wedge \tilde{a}_2 \neq 0 \rangle \twoheadrightarrow (r_1/r_2, \tfrac{|r_1|e_2+|r_2|e_1}{r_2 r_2 - e_2|r_2|} + 1/2\,\mathsf{ulp}(\tilde{a}_1/\tilde{a}_2))$$

Semantics of mathematical operators

From the semantics of the operands...

$$\langle \phi_1, \tilde{\phi}_1 \rangle \twoheadrightarrow (r_1, e_1) \in \mathcal{E}[\![\tilde{a}_1]\!]_{(\sigma,I)} \qquad \langle \phi_2, \tilde{\phi}_2 \rangle \twoheadrightarrow (r_2, e_2) \in \mathcal{E}[\![\tilde{a}_2]\!]_{(\sigma,I)}$$

...we build the semantics of the operator bottom-up

Division $\tilde{a}_1 \tilde{/} \tilde{a}_2$

$$\langle \phi_1 \wedge \phi_2 \wedge r_2 \neq 0, \tilde{\phi}_1 \wedge \tilde{\phi}_2 \wedge \tilde{a}_2 \neq 0 \rangle \twoheadrightarrow (r_1/r_2, \tfrac{|r_1|e_2+|r_2|e_1}{r_2 r_2 - e_2|r_2|} + 1/2\,\mathsf{ulp}(\tilde{a}_1/\tilde{a}_2))$$

Semantics of mathematical operators

From the semantics of the operands...

$$\langle \phi_1, \tilde{\phi}_1 \rangle \twoheadrightarrow (r_1, e_1) \in \mathcal{E}[\![\tilde{a}_1]\!]_{(\sigma,I)} \qquad \langle \phi_2, \tilde{\phi}_2 \rangle \twoheadrightarrow (r_2, e_2) \in \mathcal{E}[\![\tilde{a}_2]\!]_{(\sigma,I)}$$

...we build the semantics of the operator bottom-up

Division $\tilde{a}_1 \tilde{/} \tilde{a}_2$

$$\langle \phi_1 \wedge \phi_2 \wedge r_2 \neq 0, \tilde{\phi}_1 \wedge \tilde{\phi}_2 \wedge \tilde{a}_2 \neq 0 \rangle \twoheadrightarrow (r_1/r_2, \frac{|r_1|e_2 + |r_2|e_1}{r_2 r_2 - e_2|r_2|} + 1/2\,\mathsf{ulp}(\tilde{a}_1/\tilde{a}_2))$$

Kodiak global optimizer: from symbolic expressions to sound numerical approximations

- The information collected in the analysis is symbolic $\Rightarrow$ modular analysis

$$\mathcal{E}[\![\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y]\!]_{(\sigma,I)} = \langle true, true \rangle \twoheadrightarrow (s_x v_x + s_y v_y, \mathbf{err})$$

symbolic error expression

$$\begin{aligned} \mathbf{err} = {} & \epsilon_{\tilde{s}_x}\epsilon_{\tilde{v}_x} + |s_x|\epsilon_{\tilde{v}_x} + |v_x|\epsilon_{\tilde{s}_x} + 1/2\,\mathsf{ulp}(s_x v_x) + \\ & \epsilon_{\tilde{s}_y}\epsilon_{\tilde{v}_y} + |s_y|\epsilon_{\tilde{v}_y} + |v_y|\epsilon_{\tilde{s}_y} + 1/2\,\mathsf{ulp}(s_y v_y) + \\ & 1/2\,\mathsf{ulp}(s_x v_x + s_y v_y) \end{aligned}$$

Given initial ranges $s_x, v_x, s_y, v_y \in [-15000, 15000]$

$$\epsilon_{\tilde{s}_x}, \epsilon_{\tilde{v}_x}, \epsilon_{\tilde{s}_y}, \epsilon_{\tilde{v}_y} = 1/2\,\mathsf{ulp}(15000) = 9.09e-13$$

$$\max(e) = 1.14\text{e-}7$$

Kodiak global optimizer: from symbolic expressions to sound numerical approximations

- The information collected in the analysis is symbolic $\Rightarrow$ modular analysis

$$\mathcal{E}[\![\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y]\!]_{(\sigma,I)} = \langle true, true \rangle \twoheadrightarrow (s_x v_x + s_y v_y, \mathbf{err})$$

symbolic error expression

$$\begin{aligned} \mathbf{err} = {} & \epsilon_{\tilde{s}_x}\epsilon_{\tilde{v}_x} + |s_x|\epsilon_{\tilde{v}_x} + |v_x|\epsilon_{\tilde{s}_x} + 1/2\,\mathsf{ulp}(s_x v_x) + \\ & \epsilon_{\tilde{s}_y}\epsilon_{\tilde{v}_y} + |s_y|\epsilon_{\tilde{v}_y} + |v_y|\epsilon_{\tilde{s}_y} + 1/2\,\mathsf{ulp}(s_y v_y) + \\ & 1/2\,\mathsf{ulp}(s_x v_x + s_y v_y) \end{aligned}$$

Given initial ranges $s_x, v_x, s_y, v_y \in [-15000, 15000]$

$$\epsilon_{\tilde{s}_x}, \epsilon_{\tilde{v}_x}, \epsilon_{\tilde{s}_y}, \epsilon_{\tilde{v}_y} = 1/2\,\mathsf{ulp}(15000) = 9.09e-13$$

Kodiak branch-and-bound search

$\max(e) = 1.14\text{e-}7$

numerical error value

Sound Approximation of if-then-else

Stable Test Assumption: $\mathbb{R}$ flow = $\mathbb{F}$ flow $\Rightarrow$ unsound results

Our approach: $\mathbb{R}$ flow $\neq$ $\mathbb{F}$ flow $\Rightarrow$ $\mathbb{R}$ condition $\neq$ $\mathbb{F}$ condition

Sound Approximation of if-then-else

Stable Test Assumption: $\mathbb{R}$ flow = $\mathbb{F}$ flow $\Rightarrow$ unsound results

Our approach: $\mathbb{R}$ flow $\neq$ $\mathbb{F}$ flow $\Rightarrow$ $\mathbb{R}$ condition $\neq$ $\mathbb{F}$ condition

$$if\ \tilde{\phi}\ then\ A\ else\ B$$

Stable :

$\tilde{\phi} \wedge \mathbf{R}(\tilde{\phi})$

$\neg\tilde{\phi} \wedge \neg\,\mathbf{R}(\tilde{\phi})$

propagate the info that the flows coincide

$\mathcal{E}[\![A]\!] \Downarrow_{(\mathbf{R}(\tilde{\phi}),\tilde{\phi})}$

$\mathcal{E}[\![B]\!] \Downarrow_{(\mathbf{R}(\neg\tilde{\phi}),\neg\tilde{\phi})}$

Propagation operator:

$\langle \eta, \tilde{\eta} \rangle \twoheadrightarrow (r, e) \Downarrow_{(\phi,\tilde{\phi})} = \langle \eta \wedge \phi, \tilde{\eta} \wedge \tilde{\phi} \rangle \twoheadrightarrow (r, e)$

Sound Approximation of if-then-else

Stable Test Assumption: $\mathbb{R}$ flow = $\mathbb{F}$ flow $\Rightarrow$ unsound results

Our approach: $\mathbb{R}$ flow $\neq$ $\mathbb{F}$ flow $\Rightarrow$ $\mathbb{R}$ condition $\neq$ $\mathbb{F}$ condition

$$if\ \tilde{\phi}\ then\ A\ else\ B$$

Stable :

$\tilde{\phi} \wedge \mathbf{R}(\tilde{\phi})$ $\quad \mathcal{E}[\![A]\!] \Downarrow_{(\mathbf{R}(\tilde{\phi}), \tilde{\phi})}$

$\neg\tilde{\phi} \wedge \neg\mathbf{R}(\tilde{\phi})$ $\quad \mathcal{E}[\![B]\!] \Downarrow_{(\mathbf{R}(\neg\tilde{\phi}), \neg\tilde{\phi})}$

Unstable :

flows don't coincide

$\tilde{\phi} \wedge \neg\mathbf{R}(\tilde{\phi})$ $\quad \langle \eta \wedge \neg\mathbf{R}(\tilde{\phi}), \tilde{\eta} \wedge \tilde{\phi} \rangle \twoheadrightarrow (r_B, e_A + |r_A - r_B|)$

$\neg\tilde{\phi} \wedge \mathbf{R}(\tilde{\phi})$ $\quad \langle \eta \wedge \mathbf{R}(\tilde{\phi}), \tilde{\eta} \wedge \neg\tilde{\phi} \rangle \twoheadrightarrow (r_A, e_B + |r_A - r_B|)$

Sound Approximation of if-then-else

Stable Test Assumption: $\mathbb{R}$ flow = $\mathbb{F}$ flow $\Rightarrow$ unsound results

Our approach: $\mathbb{R}$ flow $\neq$ $\mathbb{F}$ flow $\Rightarrow$ $\mathbb{R}$ condition $\neq$ $\mathbb{F}$ condition

$$if\ \tilde{\phi}\ then\ A\ else\ B$$

Stable :

$\tilde{\phi} \wedge \mathbf{R}(\tilde{\phi})$ $\quad$ $\mathcal{E}[\![A]\!] \Downarrow_{(\mathbf{R}(\tilde{\phi}),\tilde{\phi})}$

$\neg\tilde{\phi} \wedge \neg\,\mathbf{R}(\tilde{\phi})$ $\quad$ $\mathcal{E}[\![B]\!] \Downarrow_{(\mathbf{R}(\neg\tilde{\phi}),\neg\tilde{\phi})}$

Unstable :

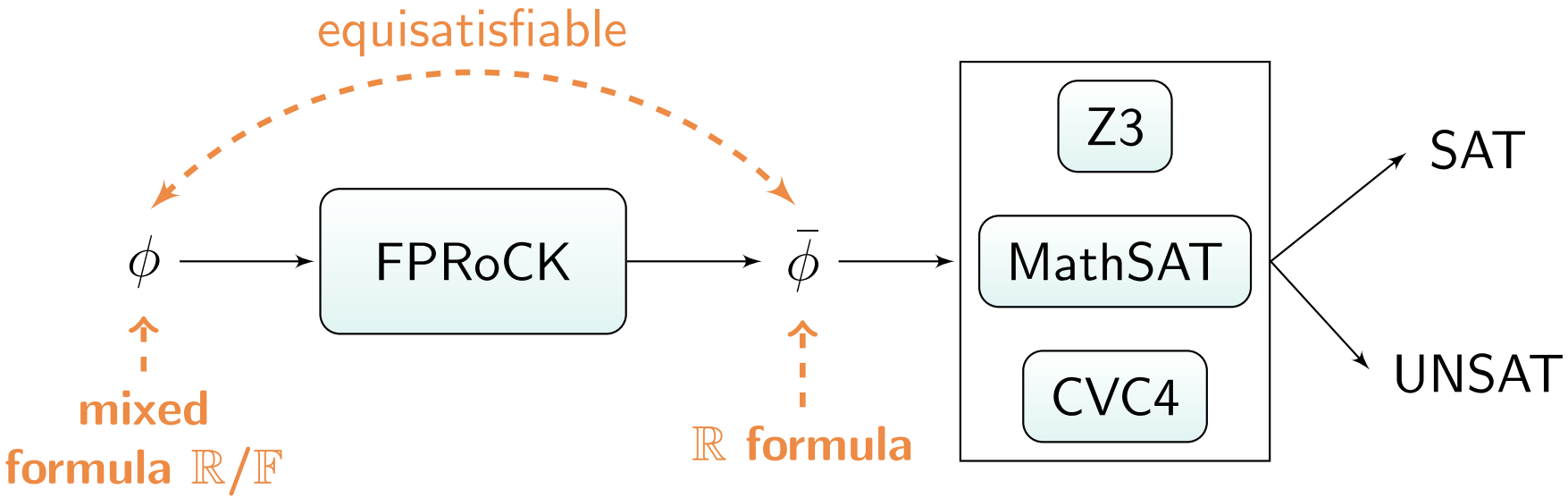
$\tilde{\phi} \wedge \neg\,\mathbf{R}(\tilde{\phi})$ $\quad$ $\langle \eta \wedge \neg\,\mathbf{R}(\tilde{\phi}), \tilde{\eta} \wedge \tilde{\phi} \rangle \twoheadrightarrow (r_B, e_A + |r_A - r_B|)$

$\neg\tilde{\phi} \wedge \mathbf{R}(\tilde{\phi})$ $\quad$ $\langle \eta \wedge \mathbf{R}(\tilde{\phi}), \tilde{\eta} \wedge \neg\tilde{\phi} \rangle \twoheadrightarrow (r_A, e_B + |r_A - r_B|)$

error of taking the incorrect branch

FPRoCK: eliminating spurious conditional error bounds

- $\langle \phi, \tilde{\phi} \rangle_{\mathbf{t}} \nrightarrow (r, e)$ is spurious? $\phi \wedge \tilde{\phi}$ is unsat?
- **Problem**: $\phi \wedge \tilde{\phi}$ has both $\mathbb{R}$ and $\mathbb{F}$ components
- FPRoCK: a solver for mixed $\mathbb{R}/\mathbb{F}$ formulas



- If $\phi \wedge \tilde{\phi}$ is UNSAT, the corresponding path condition is unfeasible
  $\Rightarrow \langle \phi, \tilde{\phi} \rangle_{\mathbf{t}} \nrightarrow (r, e)$ is dropped from the solution of PRECiSA

Example: *eps_line*

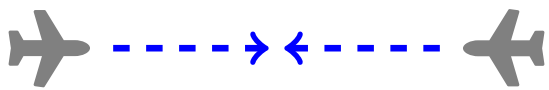
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq 0)\ then\ 1\ \ else\ -1$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

- due to round-off errors the resolution can be incorrect and the aircraft could collide



Example: *eps_line*

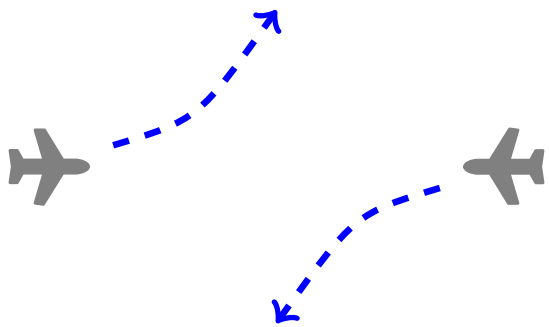
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq 0)\ then\ 1\ \ else\ -1$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

- due to round-off errors the resolution can be incorrect and the aircraft could collide



Example: *eps_line*

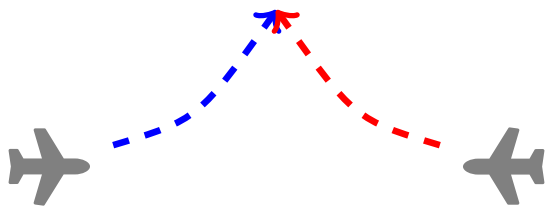
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq 0)\ then\ 1\ \ else\ -1$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

- due to round-off errors the resolution can be incorrect and the aircraft could collide



Example: *eps_line*

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq \epsilon)\ then\ 1\ elsif\ (\tilde{x} < -\epsilon)\ then\ -1\ else\ \text{error}$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

Example: *eps_line*

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq \epsilon)\ then\ 1\ elsif\ (\tilde{x} < -\epsilon)\ then\ -1\ else\ \text{error}$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

over-approx round-off error

$$|x - \tilde{x}| \leq \epsilon$$

Example: *eps_line*

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq \epsilon)\ then\ 1\ elsif\ (\tilde{x} < -\epsilon)\ then\ -1\ else\ \text{error}$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

$$\mathcal{E}[\![sign(\tilde{x})]\!]^{\pi}_{(\sigma,I)} =$$

$$\langle x \geq \epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (1, 0)$$
$$\langle x < -\epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (-1, 0)$$
$$\langle \neg x \geq \epsilon \wedge \neg x < -\epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}, 0)$$

$$\langle x \geq \epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (1, \mathbf{2})$$
$$\langle x \geq \epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}, 0)$$
$$\vdots$$
$$\langle \neg x \geq \epsilon \wedge x < -\epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (-1, \mathbf{2})$$

- Overall error $sign$ and $eps_line = 2$

Example: *eps_line*

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq \epsilon)\ then\ 1\ elsif\ (\tilde{x} < -\epsilon)\ then\ -1\ else\ \text{error}$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

$$\mathcal{E}[\![sign(\tilde{x})]\!]^{\pi}_{(\sigma,I)} =$$

$\langle x \geq \epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (1, 0)$

$\langle x < -\epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (-1, 0)$

$\langle \neg x \geq \epsilon \wedge \neg x < -\epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}$

$x \geq \epsilon \wedge \tilde{x} < -\epsilon \wedge |x - \tilde{x}| \leq \epsilon$ is UNSAT

$\langle x \geq \epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (1, \mathbf{2})$

$\langle x \geq \epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}, 0)$

$\vdots$

$\langle \neg x \geq \epsilon \wedge x < -\epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (-1, \mathbf{2})$

$\neg x \geq \epsilon \wedge \neg x < -\epsilon \wedge \tilde{x} \geq \epsilon \wedge |x - \tilde{x}| \leq \epsilon$ is UNSAT

- Overall error *sign* and *eps_line* = 2

Example: *eps_line*

eps_line

$$sign(\tilde{x}) = if\ (\tilde{x} \geq \epsilon)\ then\ 1\ elsif\ (\tilde{x} < -\epsilon)\ then\ -1\ else\ \text{error}$$

$$eps_line(\tilde{s}_x, \tilde{s}_y, \tilde{v}_x, \tilde{v}_y) = sign((\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y))$$

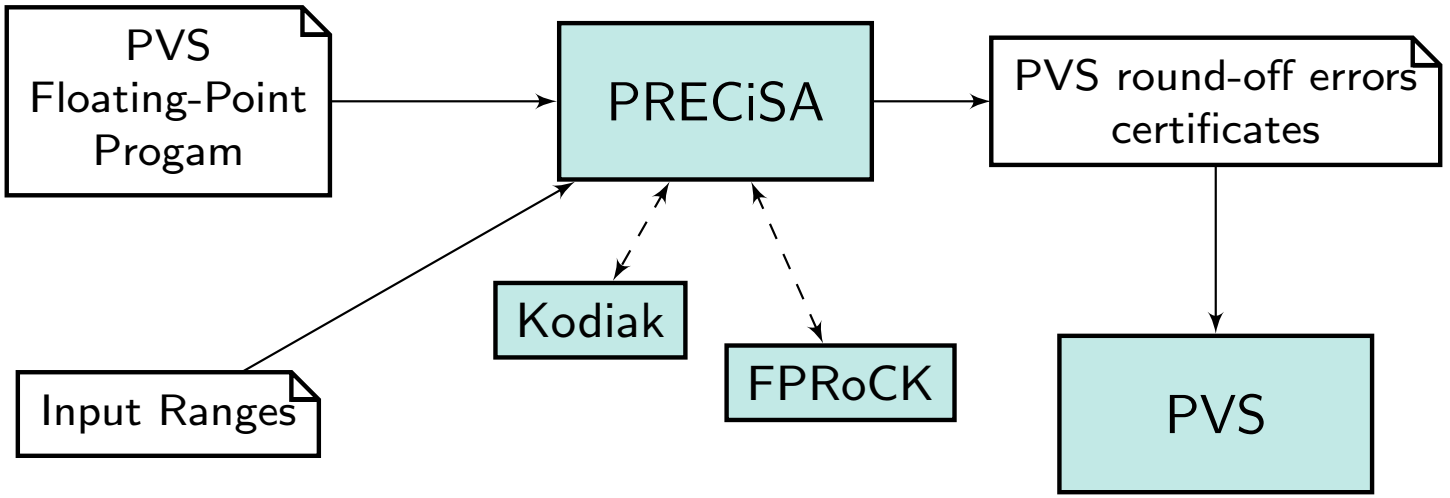
$$\mathcal{E}[\![sign(\tilde{x})]\!]^{\pi}_{(\sigma,I)} =$$

$$\langle x \geq \epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (1, 0)$$
$$\langle x < -\epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (-1, 0)$$
$$\langle \neg x \geq \epsilon \wedge \neg x < -\epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}, 0)$$

$$\langle x \geq \epsilon, \tilde{x} < -\epsilon \rangle \twoheadrightarrow (1, \mathbf{2})$$
$$\langle x \geq \epsilon, \neg \tilde{x} \geq \epsilon \wedge \neg \tilde{x} < -\epsilon \rangle \twoheadrightarrow (\text{error}, 0)$$
$$\vdots$$
$$\langle \neg x \geq \epsilon \wedge x < -\epsilon, \tilde{x} \geq \epsilon \rangle \twoheadrightarrow (-1, \mathbf{2})$$

- Overall error $sign$ and $eps_line = \not{2}\ 0$

PRECiSA-Round-off Error Static Analysis



- PRECiSA supports basic arithmetic operators, trigonometric functions, log, exp, let-in expressions, conditionals, and function calls
- abstraction α to avoid the combinatorial explosion due to nested conditionals
- widening ∇_k to converge for recursive programs
- PVS certificates ensuring the correctness of the bounds
- FPRoCK checks for UNSAT conditions that can be discarded

VSCode-PRECiSA

- VSCode-PRECiSA is a toolkit that integrates PRECiSA in Visual Studio Code
- User-friendly graphical interface
- It provides different analysis views:
  - sensitivity analysis
  - interval analysis
  - comparative analysis
  - unstable conditionals analysis

VSCode-PRECiSA: Demo

nl_comp.pvs > nl_comp

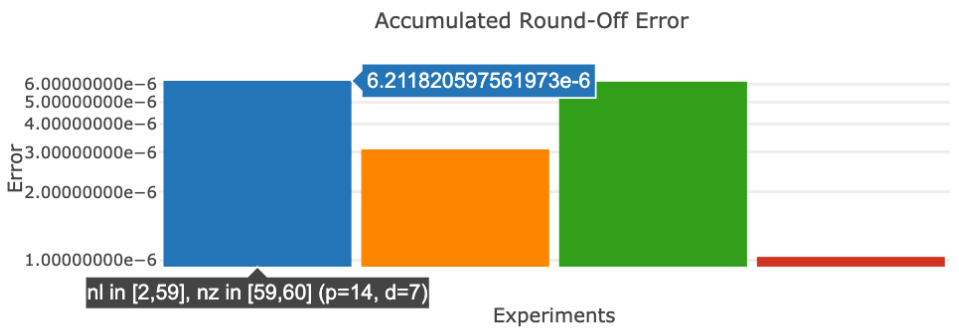
typecheck-file | evaluate-in-pvsio

```
1  nl_comp: THEORY
2
3  BEGIN
4
5  IMPORTING float@aerr754dp
6
7  % @fp-function
8  % @fp-range nl in [2,59], nz in [59,60]
   estimate-error-bounds | compare-error-bounds
9  nl_comp(nl, nz: double): double =
10   (180/3.14) *
11       acos(sqrt((1-cos(3.14/(2*nz)))
12                 /(1-cos(2*3.14/nl))))
13
14 END nl_comp
```

INPUT	MIN	MAX
nl	2	59
nz	59	60

Analyze

Save Results



Experiment	Accumulated Round-Off Error
nl in [2,59], nz in [59,60] (p=14, d=7)	6.211820597561973e-6
nl in [2,30], nz in [59,60] (p=14, d=7)	3.0859761662809845e-6
nl in [30,59], nz in [60,60] (p=14, d=7)	6.155897062805878e-6
nl in [2,10], nz in [60,60] (p=14, d=7)	1.0310720739119004e-6

Conclusions

PRECiSA is tool to analyze the rounding error of floating-point programs

- round-off error bounds + formal proof of their correctness
- PRECiSA is compositional, fully-automatic, rigorous
- Graphical user interface integrated in VSCode
- NASA Open Source Agreement (`http://github.com/nasa/precisa`)
- PRECiSA also generates stable C code from a PVS real specification instrumented to detect unstable guards → separate talk (Mariano Moscato)

References

- Mariano M. Moscato, Laura Titolo, Aaron Dutle, César A. Muñoz: *Automatic Estimation of Verified Floating-Point Round-Off Errors via Static Analysis*. SAFECOMP 2017
- Laura Titolo, Marco A. Feliú, Mariano M. Moscato, César A. Muñoz: *An Abstract Interpretation Framework for the Round-Off Error Analysis of Floating-Point Programs*. VMCAI 2018
- Rocco Salvia, Laura Titolo, Marco A. Feliú, Mariano M. Moscato, César A. Muñoz, Zvonimir Rakamaric: *A Mixed Real and Floating-Point Solver*. NFM 2019
- Paolo Masci, Laura Titolo, Marco A. Feliú, Mariano M. Moscato, Aaron Dutle: *VSCode-PRECiSA: a Toolkit for Floating-Point Round-Off Error Analysis* (to be submitted)

Thanks for your attention!