Automatic Generation of Guard-Stable Floating-Point Code

Laura Titolo[1] Mariano Moscato[1] Marco A. Feliu[1] César A. Muñoz[2]

[1]National Institute of Aerospace

[2]NASA Langley Research Center

16th International Conference on integrated Formal Methods (iFM 2020)
16-20 November 2020

Writing correct FP code is challenging

- Floating-point numbers $\mathbb{F}$ = finite representation of Reals $\mathbb{R}$
- Round-off errors $\Rightarrow$ computed $\mathbb{F}$ result $\neq$ expected $\mathbb{R}$ result
- Unstable guards = $\mathbb{F}$ control-flow $\neq$ $\mathbb{R}$ control-flow

$\Rightarrow$ Difficult to predict how round-off errors will affect the result

$\Rightarrow$ Big divergence between real and FP results in the presence of unstable guards

$\Rightarrow$ Catastrophic consequences in safety-critical software:

- Air traffic conflict avoidance systems $\Rightarrow$ resolution maneuvers that are not implicitly coordinated
- Geofencing in autonomous UAS $\Rightarrow$ incorrect determination of being inside/outside a geofence

Writing correct FP code is challenging

- Floating-point numbers $\mathbb{F}$ = finite representation of Reals $\mathbb{R}$
- Round-off errors $\Rightarrow$ computed $\mathbb{F}$ result $\neq$ expected $\mathbb{R}$ result
- Unstable guards = $\mathbb{F}$ control-flow $\neq$ $\mathbb{R}$ control-flow

$\Rightarrow$ Difficult to predict how round-off errors will affect the result

$\Rightarrow$ Big divergence between real and FP results in the presence of unstable guards

$\Rightarrow$ Catastrophic consequences in safety-critical software:

  - Air traffic conflict avoidance systems $\Rightarrow$ resolution maneuvers that are not implicitly coordinated
  - Geofencing in autonomous UAS $\Rightarrow$ incorrect determination of being inside/outside a geofence

Our solution

- Integrate three formal methods tools
  - PRECiSA: framework for the analysis of floating-point programs
  - PVS: interactive theorem prover
  - Frama-C: static analysis suite for C
- Automatically generate and verify a floating-point C implementation from a PVS real numbers specification which is
  - instrumented to detect unstable tests
  - annotated with information about round-off errors that may occur

From real-valued specifications to FP implementations

Real Specification

$$P = if\ \ x * y \geq z$$
$$then\ \ 1$$
$$else\ \ -1$$

From real-valued specifications to FP implementations

Real Specification

$$P = if\ \ x * y \geq z \quad then\ \ 1 \quad else\ \ -1$$

↓ Real2Float

Floating-Point Implementation

$$\widetilde{P} = if\ \ \tilde{x}\tilde{*}\tilde{y} \geq \tilde{z} \quad then\ \ 1 \quad else\ \ -1$$

- $\tilde{x}\tilde{*}\tilde{y} \geq \tilde{z}$ may evaluate differently from $x * y \geq z$ due to round-off errors
- the divergence is $|P - \widetilde{P}| \leq |1 - (-1)| = 2$

Program transformation to detect unstable guards

- Program transformation τ that replaces the guards in the conditionals with more restrictive ones

if $\tilde{x} \tilde{*} \tilde{y} \geq \tilde{z}$
  then 1
  else -1

$\xrightarrow{\quad\tau\quad}$

if $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} \geq \epsilon$
  then 1
elsif $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} < -\epsilon$
  then -1
  else ω

- If $\tau(P)$ does not return a *warning* ω
  - $\Rightarrow$ P returns the same value
  - $\Rightarrow$ P's execution is stable
- If P's execution is unstable
  - $\Rightarrow$ $\tau(P)$ returns a *warning* ω
- Over-approximation $\Rightarrow$ false alarms

Program transformation to detect unstable guards

- Program transformation τ that replaces the guards in the conditionals with more restrictive ones

if $\tilde{x} \tilde{*} \tilde{y} \geq \tilde{z}$
  then 1
  else -1

$\xrightarrow{\quad \tau \quad}$

if $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} \geq \epsilon$
  then 1
elsif $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} < -\epsilon$
  then -1
  else ω

over-approx round-off error of $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z}$

- If $\tau(P)$ does not return a *warning* ω
  - $\Rightarrow$ P returns the same value
  - $\Rightarrow$ P's execution is stable
- If P's execution is unstable
  - $\Rightarrow$ $\tau(P)$ returns a *warning* ω
- Over-approximation $\Rightarrow$ false alarms

Program transformation to detect unstable guards

- Program transformation τ that replaces the guards in the conditionals with more restrictive ones

if $\tilde{x} \tilde{*} \tilde{y} \geq \tilde{z}$
  then 1
  else -1

$\xrightarrow{\quad\tau\quad}$

if $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} \geq \epsilon$
  then 1
elsif $\tilde{x} \tilde{*} \tilde{y} \tilde{-} \tilde{z} < -\epsilon$
  then -1
  else ω

- If $\tau(P)$ does not return a *warning* ω
  - $\Rightarrow$ P returns the same value
  - $\Rightarrow$ P's execution is stable
- If P's execution is unstable
  - $\Rightarrow$ $\tau(P)$ returns a *warning* ω
- Over-approximation $\Rightarrow$ false alarms

Example: *eps_line*

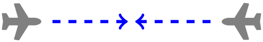
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

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 make the wrong determination of direction



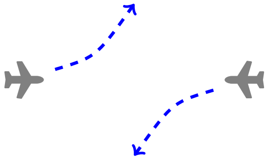
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

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 make the wrong determination of direction



Example: *eps_line*

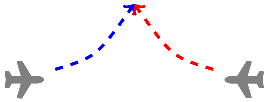
- fragment of the collision detection and avoidance algorithm *cd2d*
- used to determine implicitly coordinated horizontal resolution maneuvers to resolve air traffic conflicts

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 make the wrong determination of direction



Example: transformed *eps_line*

eps_line

$$sign(\tilde{x}, e_x) = if\ (\tilde{x} \geq e_x)\ then\ 1\ elsif\ (\tilde{x} < -e_x)\ then\ -1\ else\ \omega$$

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

over-approx
round-off error
$|x - \tilde{x}| \leq e_x$

Example: transformed *eps_line*

eps_line

$$sign(\tilde{x}, e_x) = if\ (\tilde{x} \geq e_x)\ then\ 1\ elsif\ (\tilde{x} < -e_x)\ then\ -1\ else\ \omega$$

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

new parameters

e_x **= round-off error of** x

e **= round-off error of**

$(\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{+} \tilde{s}_y \tilde{*} \tilde{v}_y) \tilde{*} (\tilde{s}_x \tilde{*} \tilde{v}_x \tilde{-} \tilde{s}_y \tilde{*} \tilde{v}_y)$

Example: transformed *eps_line*

eps_line

$$sign(\tilde{x}, e_x) = if\ (\tilde{x} \geq e_x)\ then\ 1\ elsif\ (\tilde{x} < -e_x)\ then\ -1\ else\ \omega$$

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

- Overall error $sign$ and $eps\_line = \cancel{2}$ 0
- A warning ω is issued when $\mathbb{R}$ and $\mathbb{F}$ flows diverge $(-e_x \leq \tilde{x} < e_x)$

From a real-valued PVS specification to FP C code

P = PVS **Real** specification

P = PVS **Real** specification

↓ Real2Float

$\widetilde{P}$ = **Floating-Point** implementation of P

From a real-valued PVS specification to FP C code

P = PVS **Real** specification

$\downarrow$ Real2Float

$\widetilde{P}$ = **Floating-Point** implementation of P

$\downarrow$ τ

$\widetilde{P}^{\tau}$ = **Instrumented** version of P detecting unstable tests

P = PVS **Real** specification

↓ Real2Float

$\widetilde{P}$ = **Floating-Point** implementation of P

↓ τ

$\widetilde{P}^\tau$ = **Instrumented** version of P detecting unstable tests

↓ C code generation

C implementation of $\widetilde{P}^\tau$ with **ACSL annotations**

Example: tcoa

- $tcoa$ is used in the library DAIDALUS (Detect-and-avoid) to compute the time to co-altitude of two aircraft

Time to co-altitude

$$tcoa(s_z, v_z) = if\ s_z v_z < 0\ then\ -(s_z/v_z)\ else\ 0$$

Transformed Time to co-altitude

$$\widetilde{tcoa}^{\tau}(\tilde{s}_z, \tilde{v}_z, e_{tcoa}) = \begin{array}{l} if\ \tilde{s}_z\tilde{v}_z < -e_{tcoa}\ then\ -(\widetilde{\tilde{s}/\tilde{v}}) \\ elsif\ \tilde{s}\tilde{v} \geq e_{tcoa}\ then\ 0\ \ else\ \omega \end{array} \qquad \% |(\tilde{s}_z\tilde{v}_z) - (s_z v_z)| \leq e_{tcoa}$$

$/ * @\ real\ \ tcoa(real\ \ s_z, real\ \ v_z) = s_z * v_z < 0\ ?\ -(s_z/v_z) : 0$

$double\ \ fp\_tcoa(double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) = \tilde{s}_z \tilde{*} \tilde{v}_z < 0\ ?\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z) : 0$

$predicate\ \ tcoa\_stable\_paths(real\ \ s_z, realv_z, double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) =$

$(v_z \neq 0 \wedge s_z * v_z < 0 \wedge \tilde{v}_z \neq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z < 0) \vee (s_z * v_z \geq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z \geq 0)$

$\textbf{requires} : 0 \leq e$

$\textbf{ensures} : result \neq \omega \Rightarrow (result = fp\_tcoa(\tilde{s}_z, \tilde{v}_z)$

$\wedge \forall s_z, v_z (|(\tilde{s}_z \tilde{*} \tilde{v}_z) - (s_z * v_z)| \leq e \Rightarrow tcoa\_stable\_paths(s_z, v_z, \tilde{s}_z, \tilde{v}_z))$

$* /$

$double\ \ tau\_tcoa\ (double\ \ \tilde{s}_z, double\ \ \tilde{v}_z, double\ \ e)\{$

$if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z < -e)\{$

$return\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z);$

$\}\ else\ \{\ if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z \geq -e)$

$\{return\ \ 0;$

$\}\ else\ \{return\ \ \omega;\}\}\}$

transformed program

$/{*}@\ real\ \ tcoa(real\ \ s_z, real\ \ v_z) = s_z * v_z < 0\,?\, -(s_z/v_z) : 0$

$double\ \ fp\_tcoa(double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) = \tilde{s}_z \tilde{*} \tilde{v}_z < 0\,?\, \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z) : 0$

$predicate\ \ tcoa\_stable\_paths(real\ \ s_z, realv_z, double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) =$

$(v_z \neq 0 \wedge s_z * v_z < 0 \wedge \tilde{v}_z \neq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z < 0) \vee (s_z * v_z \geq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z \geq 0)$

$\textbf{requires} : 0 \leq e$

$\textbf{ensures} : result \neq \omega \Rightarrow (result = fp\_tcoa(\tilde{s}_z, \tilde{v}_z)$

$\wedge \forall s_z, v_z (|(\tilde{s}_z \tilde{*} \tilde{v}_z) - (s_z * v_z)| \leq e \Rightarrow tcoa\_stable\_paths(s_z, v_z, \tilde{s}_z, \tilde{v}_z))$

$*/$

$double\ \ tau\_tcoa\ (double\ \ \tilde{s}_z, double\ \ \tilde{v}_z, double\ \ e)\{$

$if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z < -e)\{$

$return\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z);$

$\}\ else\ \{\ if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z \geq -e)$

$\{return\ \ 0;$

$\}\ else\ \{return\ \ \omega;\}\}\}$

real-valued specification

C code generation: symbolic function

$/{*}@\ real\ tcoa(real\ s_z, real\ v_z) = s_z * v_z < 0\ ?\ -(s_z/v_z) : 0$

$\quad double\ fp\_tcoa(double\ \tilde{s}_z, double\ \tilde{v}_z) = \tilde{s}_z \tilde{*} \tilde{v}_z < 0\ ?\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z) : 0$

$\quad predicate\ tcoa\_stable\_paths(real\ s_z, realv_z, double\ \tilde{s}_z, double\ \tilde{v}_z) =$

$\qquad (v_z \neq 0 \wedge s_z * v_z < 0 \wedge \tilde{v}_z \neq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z < 0) \vee (s_z * v_z \geq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z \geq 0)$

$\quad$ **requires** $: 0 \leq e$

$\quad$ **ensures** $: result \neq \omega \Rightarrow (result = fp\_tcoa(\tilde{s}_z, \tilde{v}_z)$

$\quad \wedge \forall s_z, v_z (|(\tilde{s}_z \tilde{*} \tilde{v}_z) - (s_z * v_z)| \leq e \Rightarrow tcoa\_stable\_paths(s_z, v_z$

floating-point version of the specification

$*/$

$double\ tau\_tcoa\ (double\ \tilde{s}_z, double\ \tilde{v}_z, double\ e)\{$

$\quad if\ (\tilde{s}_z \tilde{*} \tilde{v}_z < -e)\{$

$\quad return\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z);$

$\quad \}\ else\ \{\ if\ (\tilde{s}_z \tilde{*} \tilde{v}_z \geq -e)$

$\qquad \{return\ \ 0;$

$\qquad \}\ else\ \{return\ \ \omega;\}\}\}$

$/*@\ real\ \ tcoa(real\ \ s_z, real\ \ v_z) = s_z * v_z < 0\ ?\ -(s_z/v_z) : 0$

$double\ \ fp\_tcoa(double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) = \tilde{s}_z \tilde{*} \tilde{v}_z < 0\ ?\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z) : 0$

$predicate\ \ tcoa\_stable\_paths(real\ \ s_z, realv_z, double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) =$

$(v_z \neq 0 \wedge s_z * v_z < 0 \wedge \tilde{v}_z \neq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z < 0) \vee (s_z * v_z \geq 0 \wedge \tilde{s}_z \tilde{*} \tilde{v}_z \geq 0)$

requires : $0 \leq e$

ensures :$result \neq \omega \Rightarrow (result = fp\_tcoa(\tilde{s}_z, \tilde{v}_z)$

$v_z(|(\tilde{s}_z \tilde{*} \tilde{v}_z) - (s_z * v_z)| \leq e \Rightarrow tcoa\_stable\_paths(s_z, v_z, \tilde{s}_z, \tilde{v}_z))$

stable paths predicate

$double\ \ tau\_tcoa\ (double\ \ \tilde{s}_z, double\ \ \tilde{v}_z, double\ \ e)\{$

$if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z < -e)\{$

$return\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z);$

$\}\ else\ \{\ if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z \geq -e)$

$\{return\ \ 0;$

$\}\ else\ \{return\ \ \omega;\}\}\}$

C code generation: symbolic function

$$
\begin{aligned}
&/*@\ real\ \ tcoa(real\ \ s_z, real\ \ v_z) = s_z * v_z < 0\ ?\ -(s_z/v_z) : 0\\
&\quad double\ \ fp\_tcoa(double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) = \tilde{s}_z \tilde{*} \tilde{v}_z < 0\ ?\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z) : 0\\
&\quad predicate\ \ tcoa\_stable\_paths(real\ \ s_z, realv_z, double\ \ \tilde{s}_z, double\ \ \tilde{v}_z) =\\
&\qquad (v_z \neq 0 \land s_z * v_z < 0 \land \tilde{v}_z \neq 0 \land \tilde{s}_z \tilde{*} \tilde{v}_z < 0) \lor (s_z * v_z \geq 0 \land \tilde{s}_z \tilde{*} \tilde{v}_z \geq 0)\\
&\quad \textbf{requires} : 0 \leq e\\
&\quad \textbf{ensures} : result \neq \omega \Rightarrow (result = fp\_tcoa(\tilde{s}_z, \tilde{v}_z)\\
&\quad \land \forall s_z, v_z (|(\tilde{s}_z \tilde{*} \tilde{v}_z) - (s_z * v_z)| \leq e \Rightarrow tcoa\_stable\_paths(s_z, v_z, \tilde{s}_z, \tilde{v}_z))\\
&\ \ */\\
&double\ \ tau\_tcoa\ (double\ \ \tilde{s}_z, double\ \ \tilde{v}_z, double\ \ e)\{\\
&\quad if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z < -e)\{\\
&\quad return\ \tilde{-}(\tilde{s}_z \tilde{/} \tilde{v}_z);\\
&\quad \}\ else\ \{\ if\ \ (\tilde{s}_z \tilde{*} \tilde{v}_z \geq -e)\\
&\qquad\qquad \{return\ \ \ 0;\\
&\qquad\qquad \}\ else\ \{return\ \ \ \omega;\}\}\}
\end{aligned}
$$

post-condition

C code generation: numeric function

- symbolic functions do not depend on initial ranges for the input vars
- PRECiSA uses the global optimizer Kodiak to maximize the symbolic error expression given these ranges

$$
\begin{aligned}
/{*}@\mathbf{ensures}: &\forall s_z, v_z (1 \le s_z \le 1000 \wedge 1 \le v_z \le 1000 \wedge \\
&|\tilde{s}_z - s_z| \le ulp(s_z)/2 \wedge |\tilde{v}_z - v_z| \le ulp(v_z)/2) \wedge \\
&result \ne \omega \\
&\Rightarrow |result - tcoa(s_z, v_z)| \le 2.78e-12 \\
&{*}/
\end{aligned}
$$

$$
\begin{aligned}
&double\ tau\_tcoa\_num(double\ \tilde{s}_z, double\ \tilde{v}_z)\{ \\
&\quad return\ \ tau\_tcoa\ (\tilde{s}_z, \tilde{v}_z, 1.72e-10)\}
\end{aligned}
$$

call to symbolic function

round-off error e computed by Kodiak

C code generation: numeric function

- symbolic functions do not depend on initial ranges for the input vars
- PRECiSA uses the global optimizer Kodiak to [initial values arguments] symbolic error expression given these ranges

initial values arguments

$$/{*}\text{@}\mathbf{ensures}: \forall s_z, v_z (1 \le s_z \le 1000 \wedge 1 \le v_z \le 1000 \wedge$$
$$|\tilde{s}_z - s_z| \le ulp(s_z)/2 \wedge |\tilde{v}_z - v_z| \le ulp(v_z)/2) \wedge$$
$$result \neq \omega$$
$$\Rightarrow |result - tcoa(s_z, v_z)| \le 2.78e-12$$
$$*/$$

round-off errors arguments

$$double\ tau\_tcoa\_num(double\ \tilde{s}_z, double\ \tilde{v}_z)\{$$
$$return\ \ tau\_tcoa\ (\tilde{s}_z, \tilde{v}_z, 1.72e-10)\}$$

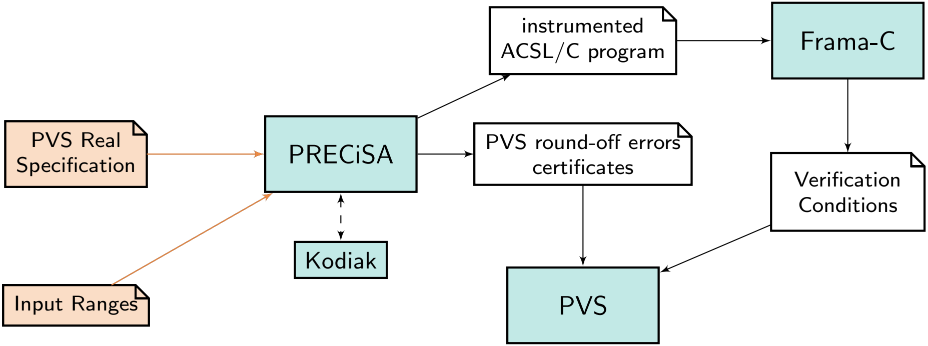
- symbolic functions do not depend on initial ranges for the input vars
- PRECiSA uses the global optimizer Kodiak to maximize the symbolic error expression given these ranges

$$
\begin{aligned}
/{*}@\textbf{ensures}: &\forall s_z, v_z (1 \leq s_z \leq 1000 \wedge 1 \leq v_z \leq 1000 \wedge \\
&|\tilde{s}_z - s_z| \leq ulp(s_z)/2 \wedge |\tilde{v}_z - v_z| \leq ulp(v_z)/2) \wedge \\
&result \neq \omega \\
&\Rightarrow |result - tcoa(s_z, v_z)| \leq 2.78e-12 \\
&{*}/
\end{aligned}
$$

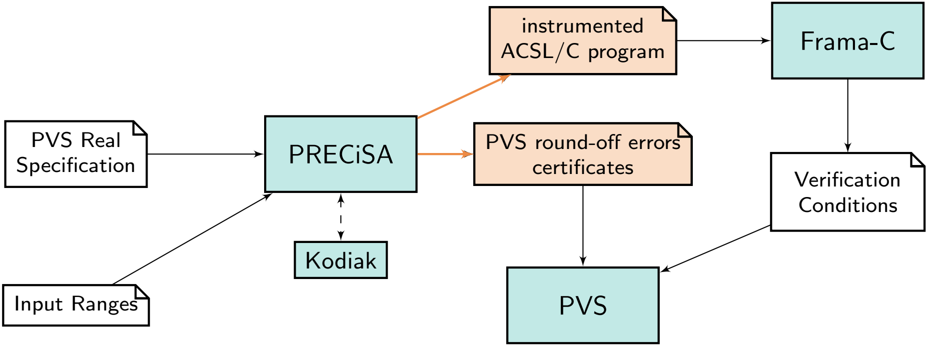
$$
\begin{aligned}
&double\ tau\_tcoa\_num(double\ \tilde{s}_z, double\ \tilde{v}_z)\{ \\
&\quad return\ \ tau\_tcoa\ (\tilde{s}_z, \tilde{v}_z, 1.72e-10)\}
\end{aligned}
$$

round-off error of tau_tcoa_num computed by Kodiak

PRECiSA: stable C code generation



PRECiSA: stable C code generation



generate the C code annotated with ACSL stating the relation between $\mathbb{R}$ spec and $\mathbb{F}$ program

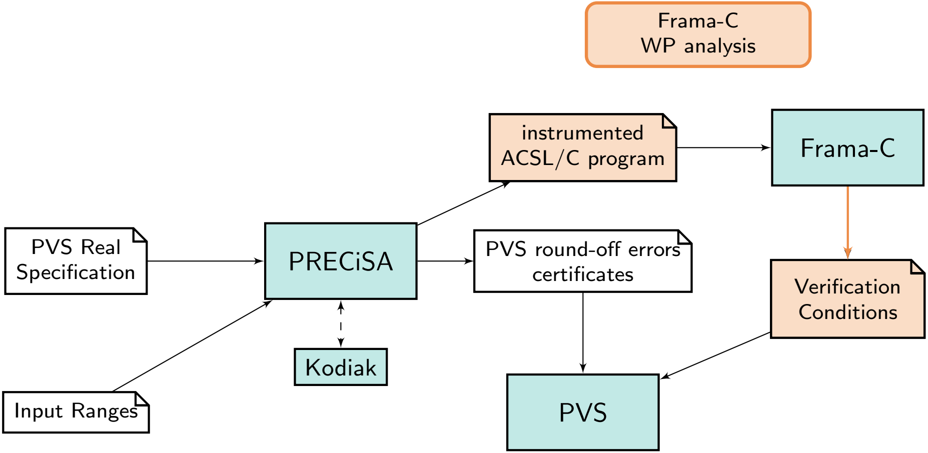
C code verification

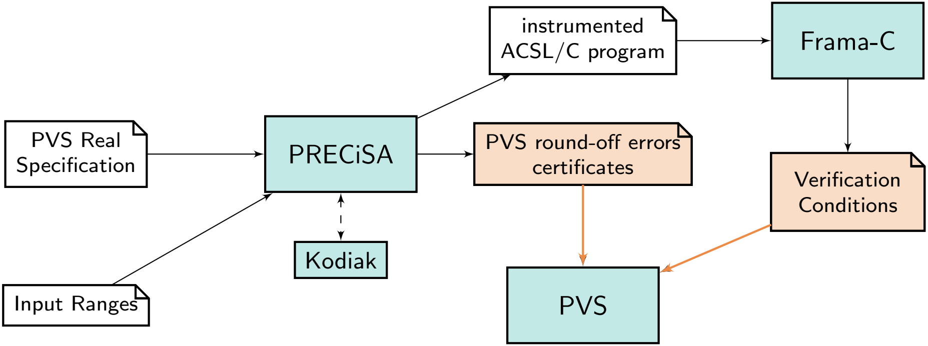
- the Frama-C/WP plug-in is used to generate VCs in the language of PVS
- customization to link the PRECiSA round-off error certificates and floating-point formalization

Verification Conditions for tcoa

$$\varphi_{tau\_tcoa} = \forall e, s_z, v_z, e_s, e_v \in \mathbb{R}, \tilde{s}, \tilde{v} \in \mathbb{F}$$
$$(result \neq \omega \wedge e \geq 0 \wedge |\tilde{v}_z - v_z| \leq e_v \wedge |\tilde{s}_z - s_z| \leq e_s \wedge |(\tilde{s}_z \tilde{*} \tilde{v}_z) - (v_z * s_z)| \leq e$$
$$\Rightarrow result = fp\_tcoa(s_z, v_z)).$$

$$\varphi_{tau\_tcoa\_num} = \forall s_z, v_z \in \mathbb{R}, \tilde{s}_z, \tilde{v}_z \in \mathbb{F}, (result \neq \omega \wedge 1 \leq \tilde{s}_z \leq 1000 \wedge 1 \leq \tilde{v}_z \leq 1000$$
$$\wedge |s_z - \tilde{s}_z| \leq \tfrac{1}{2}\, ulp(s_z) \wedge |v_z - \tilde{v}_z| \leq \tfrac{1}{2}\, ulp(v_z) \wedge |(\tilde{s}_z \tilde{*} \tilde{v}_z) - (v_z * s_z)| \leq \textit{1.72e-10})$$
$$\Rightarrow |result - tcoa(s_z, v_z)| \leq \textit{2.78e-12}$$


Frama-C
WP analysis
instrumented
ACSL/C program
Frama-C
PVS Real
Specification
PRECiSA
PVS round-off errors
certificates
Verification
Conditions
Kodiak
Input Ranges
PVS



check in PVS with automatic strategies

Conclusions

- Successful integration of three formal methods tools: PRECiSA, PVS, and Frama-C
- generate stable C code from a PVS real specification instrumented to detect unstable tests
- automatic verification with Frama-C+PVS
  $\Rightarrow$ no user expertise required in FP arithmetic or theorem proving
- NASA Open Source Agreement (`http://github.com/nasa/precisa`)
- application to significant fragments of the NASA formalizations of PolyCARP (geofencing) and DAIDALUS (detect-and-avoid)

Thanks for your attention!
Questions?