De l'arithmétique d'intervalles à la certification de programmes

Guillaume Melquiond

Sous la direction de Marc Daumas
Laboratoire de l'Informatique du Parallélisme
Arénaire, LIP, CNRS–ENSL–INRIA–UCBL

2006-11-21

From interval arithmetic to program certification

Guillaume Melquiond

Advisor: Marc Daumas
Laboratoire de l'Informatique du Parallélisme
Arénaire, LIP, CNRS–ENSL–INRIA–UCBL

2006-11-21

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,
- limited precision $\Rightarrow$ inaccurate results.

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,

  Ariane 5 maiden flight: \$500M
- limited precision $\Rightarrow$ inaccurate results.

  Patriot missile failure: 28 casualties

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,

  Ariane 5 maiden flight: $500M
- limited precision $\Rightarrow$ inaccurate results.

  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,
  Ariane 5 maiden flight: $500M
- limited precision $\Rightarrow$ inaccurate results.
  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Unfortunately, certifying a numerical application is

- long and tedious,

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,
  Ariane 5 maiden flight: \$500M
- limited precision $\Rightarrow$ inaccurate results.
  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Unfortunately, certifying a numerical application is

- long and tedious,
- error-prone.

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,
  Ariane 5 maiden flight: \$500M
- limited precision $\Rightarrow$ inaccurate results.
  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Unfortunately, certifying a numerical application is

- long and tedious,
- error-prone.

Automated tool

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,

  Ariane 5 maiden flight: $500M
- limited precision $\Rightarrow$ inaccurate results.

  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Unfortunately, certifying a numerical application is

- long and tedious, Automated tool
- error-prone. Formal methods

Motivation

Floating-point and fixed-point datatypes suffer from:

- limited range $\Rightarrow$ underflow, overflow,

  Ariane 5 maiden flight: $500M
- limited precision $\Rightarrow$ inaccurate results.

  Patriot missile failure: 28 casualties

$\Longrightarrow$ Safety-critical applications require certification.

Unfortunately, certifying a numerical application is

- long and tedious, Automated tool
- error-prone. Formal methods

$\left.\begin{array}{r}\text{Automated tool}\\ \text{Formal methods}\end{array}\right\}$ Gappa

Motto of this PhD

1. What most users identify as simple ideas should be easily usable as formal methods.
2. A computer should not require help from the user for problems that can be solved with some limited work.

Outline

Outline

Example: orientation of three points

Given three points p, q, and r of the 2D plane, they can be either aligned or clockwise-oriented or counter-clockwise-oriented.

$$\mathrm{orient}_2(p,q,r) = \mathrm{sign} \begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$

Example: orientation of three points

Given three points p, q, and r of the 2D plane, they can be either aligned or clockwise-oriented or counter-clockwise-oriented.

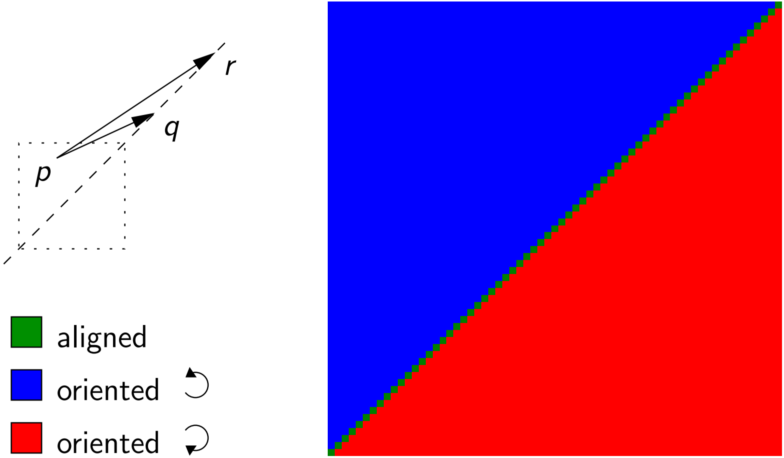
$$\mathrm{orient}_2(p, q, r) = \mathrm{sign} \begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$

A naive floating-point implementation:

```
float det = (qx - px) * (ry - py)
          - (qy - py) * (rx - px);
if (det > 0) return POSITIVE;
if (det < 0) return NEGATIVE;
return ZERO;
```

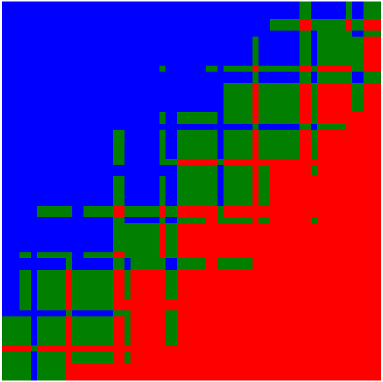
Infinitely precise computations

For $q = (8.1, 8.1)$ and $r = (12.1, 12.1)$ and p around $(1.5, 1.5)$, the sign of the determinant should look like:



Actual single-precision computations

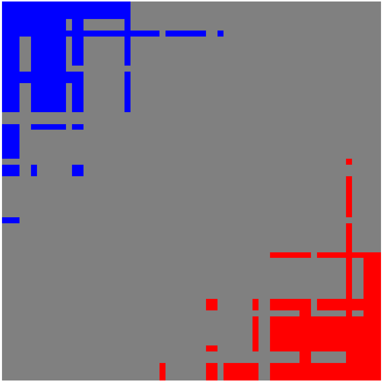
Due to the limited precision of floating-point numbers, the computed sign may be wrong. It actually looks like:



Robust computations

The computed value det and the exact value Det have the same sign when $|\mathtt{det}| > \xi$, with ξ an upper bound on $|\mathtt{det} - \mathtt{Det}|$.

Improvement: flag results that are not guaranteed to be correct.



Introducing Gappa

Computing a bound on $|\mathtt{det} - \mathtt{Det}|$ with Gappa:

```
# Single precision and round to nearest
@rnd = float< ieee_32, ne >;

# Input variables (floating-point numbers)
px = rnd(px_);   py = rnd(py_);
qx = rnd(8.1);   qy = rnd(8.1);
rx = rnd(12.1);  ry = rnd(12.1);

# Computed and exact values of the determinant
det rnd= (qx-px)*(ry-py) - (qy-py)*(rx-px);
Det   =  (qx-px)*(ry-py) - (qy-py)*(rx-px);

# Logical formula
{ |px - 1.5| <= 32b-23 /\
  |py - 1.5| <= 32b-23 ->
  |det - Det| in ? }
```

Improved algorithm

Gappa's answer: the property

$$\|p - (1.5, 1.5)\|_\infty \leq 32 \cdot 2^{-23} \quad \Longrightarrow \quad |\mathtt{det} - \mathtt{Det}| \leq \xi$$

is provable for $\xi = 1.9 \cdot 10^{-5}$.

Improved algorithm

Gappa's answer: the property

$$\|p - (1.5, 1.5)\|_\infty \leq 32 \cdot 2^{-23} \quad \Longrightarrow \quad |\mathtt{det} - \mathtt{Det}| \leq \xi$$

is provable for $\xi = 1.9 \cdot 10^{-5}$.

Gappa also generates a Coq formal proof: 1823 lines, 391 lemmas.

Improved algorithm

Gappa's answer: the property

$$\|p - (1.5, 1.5)\|_\infty \leq 32 \cdot 2^{-23} \quad \Longrightarrow \quad |\texttt{det} - \texttt{Det}| \leq \xi$$

is provable for $\xi = 1.9 \cdot 10^{-5}$.

Gappa also generates a Coq formal proof: 1823 lines, 391 lemmas.

Robust floating-point implementation:

```
float det = (qx - px) * (ry - py)
          - (qy - py) * (rx - px);
if (det > +1.9e-5) return POSITIVE;
if (det < -1.9e-5) return NEGATIVE;
return UNKNOWN;
```

The Gappa tool

Objective: help users certify/analyze their numerical applications.

Design decisions:

- the tool verifies enclosures of mathematical expressions;
- these expressions can contain rounding operators to express limitations and properties of datatypes;
- formal proofs are generated to provide confidence in the development.

The Gappa tool

Objective: help users certify/analyze their numerical applications.

Design decisions:

- the tool verifies enclosures of mathematical expressions;
- these expressions can contain rounding operators to express limitations and properties of datatypes;
- formal proofs are generated to provide confidence in the development.

How does it work?

- Interval arithmetic for propagating enclosures.
- Theorems on bounds on rounded values and rounding errors.
- Rewriting rules for tightening computed enclosures.

Outline

Model: bounding expressions by numeric intervals

Basic element: an enclosure $e \in I$.

- e is an expression on real numbers:
  $e ::= \mathit{number} \mid -e \mid \circ(e) \mid e + e \mid e \times e \mid \sqrt{e} \mid \ldots$
- $I = [a, b]$ is an interval with dyadic rational bounds.

Model: bounding expressions by numeric intervals

Basic element: an enclosure $e \in I$.

- e is an expression on real numbers:
  $$e ::= \mathit{number} \mid -e \mid \circ(e) \mid e+e \mid e \times e \mid \sqrt{e} \mid \ldots$$
- $I = [a, b]$ is an interval with dyadic rational bounds.

These enclosures are appropriate to express questions that usually arise when certifying numerical applications:

- no overflow, no invalid operations, etc
  - variable domain: $\tilde{x} \in I$,
- accuracy of computed values
  - absolute error: $\tilde{x} - x \in I$,
  - relative error: $(\tilde{x} - x)/x \in I$.

Interval arithmetic as proof foundation

Interval evaluations can serve as proofs of bounds,
when they satisfy the containment property:

$$x \in I_x \wedge y \in I_y \implies x \diamond y \in I_z \quad \text{if } I_x \diamond I_y \subseteq I_z$$

for $\diamond \in \{+, -, \times, \div\}$. Also for unary functions: $\sqrt{\cdot}, \sin, \ldots$

Interval arithmetic as proof foundation

Interval evaluations can serve as proofs of bounds, when they satisfy the containment property:

$$x \in I_x \wedge y \in I_y \quad \Longrightarrow \quad x \diamond y \in I_z \quad \text{if } I_x \diamond I_y \subseteq I_z$$

for $\diamond \in \{+, -, \times, \div\}$. Also for unary functions: $\sqrt{\cdot}, \sin, \ldots$

Arithmetic operations on intervals:

- $[a, b] + [c, d] = [a + c, b + d]$,
- $[a, b] - [c, d] = [a - d, b - c]$,
- $[a, b] \times [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$,
- $[a, b] \div [c, d] = [a, b] \times [c, d]^{-1}$
  with $[c, d]^{-1} = [1/d, 1/c]$ if $0 \notin [c, d]$.

Example: square root

Property to prove: $x \in [3, 5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.

Example: square root

Property to prove: $x \in [3, 5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a, b]$, then $\sqrt{e} \in [\sqrt{a}, \sqrt{b}]$.

Example: square root

Property to prove: $x \in [3, 5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a, b]$, then $\sqrt{e} \in [\triangledown\sqrt{a}, \vartriangle\sqrt{b}]$.

Example: square root

Property to prove: $x \in [3, 5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a, b]$, then $\sqrt{e} \in [\triangledown\sqrt{a}, \triangle\sqrt{b}]$.

Verifying the property: (in Gappa)

- $[\triangledown\sqrt{3}, \triangle\sqrt{5}] = [1773 \cdot 2^{-10}, 1145 \cdot 2^{-9}] \subseteq [1.3, 2.3]$.

Example: square root

Property to prove: $x \in [3, 5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a, b]$, then $\sqrt{e} \in [\triangledown\sqrt{a}, \triangle\sqrt{b}]$.

Verifying the property: (in Gappa)

- $[\triangledown\sqrt{3}, \triangle\sqrt{5}] = [1773 \cdot 2^{-10}, 1145 \cdot 2^{-9}] \subseteq [1.3, 2.3]$.

Checking the certificate: (in Coq)

1. $\sqrt{x} \in [1773 \cdot 2^{-10}, \ldots]$ holds
   because $(1773 \cdot 2^{-10})^2 = 3143529 \cdot 2^{-20} \leq 3 \leq x$.
2. $1.3 \leq 1773 \cdot 2^{-10}$ holds.

Example: square root

Property to prove: $x \in [3,5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a,b]$, then $\sqrt{e} \in [\triangledown\sqrt{a}, \triangle\sqrt{b}]$.

Verifying the property: (in Gappa)

- $[\triangledown\sqrt{3}, \triangle\sqrt{5}] = [1773 \cdot 2^{-10}, 1145 \cdot 2^{-9}] \subseteq [1.3, 2.3]$.

Checking the certificate: (in Coq)

1. $\sqrt{x} \in [1773 \cdot 2^{-10}, \ldots]$ holds
   because $(1773 \cdot 2^{-10})^2 = 3143529 \cdot 2^{-20} \leq 3 \leq x$.
2. $1.3 \leq 1773 \cdot 2^{-10}$ holds.

Simplifying the certificate: (in Gappa)

- $1.3 \leq \frac{3}{2} \leq 1773 \cdot 2^{-10}$ and $1145 \cdot 2^{-9} \leq \frac{9}{4} \leq 2.3$.

Example: square root

Property to prove: $x \in [3,5] \Rightarrow \sqrt{x} \in [1.3, 2.3]$.
Interval theorem: if $e \in [a,b]$, then $\sqrt{e} \in [\triangledown\sqrt{a}, \vartriangle\sqrt{b}]$.

Verifying the property: (in Gappa)

- $[\triangledown\sqrt{3}, \vartriangle\sqrt{5}] = [1773 \cdot 2^{-10}, 1145 \cdot 2^{-9}] \subseteq [1.3, 2.3]$.

Checking the certificate: (in Coq)

1. $\sqrt{x} \in [\frac{3}{2}, \ldots]$ holds because $(\frac{3}{2})^2 = 9 \cdot 2^{-2} \leq 3 \leq x$.
2. $1.3 \leq \frac{3}{2}$ holds.

Simplifying the certificate: (in Gappa)

- $1.3 \leq \frac{3}{2} \leq 1773 \cdot 2^{-10}$ and $1145 \cdot 2^{-9} \leq \frac{9}{4} \leq 2.3$.
- A certificate using $\sqrt{x} \in \left[\frac{3}{2}, \frac{9}{4}\right]$ is checked faster by Coq.

Digression: the Boost C++ library

Gappa relies on the Boost C++ interval arithmetic library.

Digression: the Boost C++ library

Gappa relies on the Boost C++ interval arithmetic library.

Generic library in the spirit of the STL:

- instantiated with `double` and GMP rationals
  for PVS proofs on approximation errors,
- instantiated with MPFR for Gappa's dyadic bounds.

Digression: the Boost C++ library

Gappa relies on the Boost C++ interval arithmetic library.

Generic library in the spirit of the STL:

- instantiated with `double` and GMP rationals
  for PVS proofs on approximation errors,
- instantiated with MPFR for Gappa's dyadic bounds.

Boost: sandbox for developing new features of the C++ language.

Digression: standardizing interval arithmetic

Proposing interval arithmetic for the ISO C++ Standard.

- Motivation: giving more exposure to reliable computing to the general C++ programming community.
- Form: a pure library, no core language change.
- Target: Technical Report 2 ($\sim$ 2010).

Digression: standardizing interval arithmetic

Proposing interval arithmetic for the ISO C++ Standard.

- Motivation: giving more exposure to reliable computing to the general C++ programming community.
- Form: a pure library, no core language change.
- Target: Technical Report 2 ($\sim$ 2010).

Usage:

```
std::interval<double>
  I(1,2), J("[3.1,4.7]"), K;
K = exp(I) - J;
std::cout << K << std::endl;
```

Gappa's process

1. Extract a logical formula

$$e_1 \in I_1 \;\wedge\; \cdots \;\wedge\; e_n \in I_n \quad\Longrightarrow\quad e_{n+1} \in I_{n+1}.$$

Gappa's process

1. Extract a logical formula

$$e_1 \in I_1 \ \wedge \ \cdots \ \wedge \ e_n \in I_n \quad \Longrightarrow \quad e_{n+1} \in I_{n+1}.$$

2. Select expressions and theorems potentially useful as intermediate steps for bounding $e_1, \cdots, e_{n+1}$.

Gappa's process

1. Extract a logical formula

$$e_1 \in I_1 \ \wedge \ \cdots \ \wedge \ e_n \in I_n \quad \Longrightarrow \quad e_{n+1} \in I_{n+1}.$$

2. Select expressions and theorems potentially useful as intermediate steps for bounding $e_1, \cdots, e_{n+1}$.

3. Assuming $e_1 \in I_1, \cdots, e_n \in I_n$, compute and recompute the ranges of all the intermediate expressions until the enclosure $e_{n+1} \in I_{n+1}$ is proved.

   Keep track of the theorems as they are applied.

Gappa's process

1. Extract a logical formula

$$e_1 \in I_1 \ \wedge \ \cdots \ \wedge \ e_n \in I_n \quad \Longrightarrow \quad e_{n+1} \in I_{n+1}.$$

2. Select expressions and theorems potentially useful as intermediate steps for bounding $e_1, \cdots, e_{n+1}$.

3. Assuming $e_1 \in I_1, \cdots, e_n \in I_n$, compute and recompute the ranges of all the intermediate expressions until the enclosure $e_{n+1} \in I_{n+1}$ is proved.

   Keep track of the theorems as they are applied.

4. Simplify the resulting proof graph.

Gappa's process

1. Extract a logical formula

$$e_1 \in I_1 \;\wedge\; \cdots \;\wedge\; e_n \in I_n \quad\Longrightarrow\quad e_{n+1} \in I_{n+1}.$$

2. Select expressions and theorems potentially useful as intermediate steps for bounding $e_1, \cdots, e_{n+1}$.

3. Assuming $e_1 \in I_1, \cdots, e_n \in I_n$, compute and recompute the ranges of all the intermediate expressions until the enclosure $e_{n+1} \in I_{n+1}$ is proved.

   Keep track of the theorems as they are applied.

4. Simplify the resulting proof graph.

5. Generate a formal certificate.

Intersection and union

Intervals are sets. We can

- Improve ranges by intersection: $e \in I_1 \wedge e \in I_2 \Rightarrow e \in I_1 \cap I_2$.

Intersection and union

Intervals are sets. We can

- Improve ranges by intersection: $e \in I_1 \wedge e \in I_2 \Rightarrow e \in I_1 \cap I_2$.

  Empty intersection implies contradiction: proof by absurd.

Intersection and union

Intervals are sets. We can

- Improve ranges by intersection: $e \in I_1 \wedge e \in I_2 \Rightarrow e \in I_1 \cap I_2$.
  Empty intersection implies contradiction: proof by absurd.
- Perform case studies:
  $(x \in I_1 \Rightarrow e \in I) \wedge (x \in I_2 \Rightarrow e \in I) \implies (x \in I_1 \cup I_2 \Rightarrow e \in I)$.

Intersection and union

Intervals are sets. We can

- Improve ranges by intersection: $e \in I_1 \wedge e \in I_2 \Rightarrow e \in I_1 \cap I_2$.
  Empty intersection implies contradiction: proof by absurd.
- Perform case studies:
  $(x \in I_1 \Rightarrow e \in I) \wedge (x \in I_2 \Rightarrow e \in I) \implies (x \in I_1 \cup I_2 \Rightarrow e \in I)$.

Example: let x and y be two integers.

Full

```
{ |x| <= 100 /\ |y| <= 100 /\ x * y <= 0
  -> |x + y| <= 100 }
```

Doable by studying cases on the signs of x and y,
and detecting contradictions with $x \cdot y \leq 0$.

(Coq proof: 395 lines, 71 lemmas)

Outline

Rounding operators

How to express approximate computations?

```
double f(double a, double b, double c)
{ return a / b + c; }
```

Rounding operators

How to express approximate computations?

```
double f(double a, double b, double c)
{ return a / b + c; }
```

IEEE-754 says: a floating-point operator shall behave as if it was first computing the infinitely precise value and then rounding it so that it fits in the destination floating-point format.

Rounding operators

How to express approximate computations?

```
double f(double a, double b, double c)
{ return a / b + c; }
```

IEEE-754 says: a floating-point operator shall behave as if it was first computing the infinitely precise value and then rounding it so that it fits in the destination floating-point format.

Approach: for a given numeric environment, define one single operator $\circ(\cdot)$ on real numbers.

$$\texttt{a / b + c} \quad \longrightarrow \quad \circ(\circ(\underbrace{a \div b}_{\text{real division}}) + c)$$

Rounded values and rounding errors

For a given rounding operator, the following theorems are provided.

- An interval extension: $e \in I \Rightarrow \circ(e) \in \circ(I)$.
  Rounding monotony: $\circ([a, b]) = [\circ(a), \circ(b)]$.

Rounded values and rounding errors

For a given rounding operator, the following theorems are provided.

- An interval extension: $e \in I \Rightarrow \circ(e) \in \circ(I)$.
  Rounding monotony: $\circ([a, b]) = [\circ(a), \circ(b)]$.
- An interval restriction: $\circ(e) \in I \Rightarrow \circ(e) \in J$.
  Example: $\lfloor e \rfloor \in [1.5, 2.5]$ implies $\lfloor e \rfloor = 2$.

Rounded values and rounding errors

For a given rounding operator, the following theorems are provided.

- An interval extension: $e \in I \Rightarrow \circ(e) \in \circ(I)$.
  Rounding monotony: $\circ([a, b]) = [\circ(a), \circ(b)]$.
- An interval restriction: $\circ(e) \in I \Rightarrow \circ(e) \in J$.
  Example: $\lfloor e \rfloor \in [1.5, 2.5]$ implies $\lfloor e \rfloor = 2$.
- Bounds on rounding errors.
  Given either $e \in I$, $|e| \in I$, $\circ(e) \in I$, or $|\circ(e)| \in I$,
  they compute the ranges of

Rounded values and rounding errors

For a given rounding operator, the following theorems are provided.

- An interval extension: $e \in I \Rightarrow \circ(e) \in \circ(I)$.
  Rounding monotony: $\circ([a, b]) = [\circ(a), \circ(b)]$.
- An interval restriction: $\circ(e) \in I \Rightarrow \circ(e) \in J$.
  Example: $\lfloor e \rfloor \in [1.5, 2.5]$ implies $\lfloor e \rfloor = 2$.
- Bounds on rounding errors.
  Given either $e \in I$, $|e| \in I$, $\circ(e) \in I$, or $|\circ(e)| \in I$, they compute the ranges of
  - the absolute error $\circ(e) - e$,

Rounded values and rounding errors

For a given rounding operator, the following theorems are provided.

- An interval extension: $e \in I \Rightarrow \circ(e) \in \circ(I)$.
  Rounding monotony: $\circ([a, b]) = [\circ(a), \circ(b)]$.
- An interval restriction: $\circ(e) \in I \Rightarrow \circ(e) \in J$.
  Example: $\lfloor e \rfloor \in [1.5, 2.5]$ implies $\lfloor e \rfloor = 2$.
- Bounds on rounding errors.
  Given either $e \in I$, $|e| \in I$, $\circ(e) \in I$, or $|\circ(e)| \in I$, they compute the ranges of
  - the absolute error $\circ(e) - e$,
  - the relative error $\frac{\circ(e)-e}{e}$.

Floating-point arithmetic

A binary floating-point value can be written $m \cdot 2^e$ with $m, e \in \mathbb{Z}$.

Floating-point arithmetic

A binary floating-point value can be written $m \cdot 2^e$ with $m, e \in \mathbb{Z}$.

An IEEE-754 format imposes

- a precision p: $|m| < 2^p$,
- a minimal exponent $E_{\min}$: $e \geq E_{\min}$,
- a maximal exponent $E_{\max}$: $e \leq E_{\max}$.

Floating-point arithmetic

A binary floating-point value can be written $m \cdot 2^e$ with $m, e \in \mathbb{Z}$.

An IEEE-754 format imposes

- a precision p: $|m| < 2^p$,
- a minimal exponent $E_{\min}$: $e \geq E_{\min}$,
- a maximal exponent $E_{\max}$: $e \leq E_{\max}$.

Floating-point arithmetic

A binary floating-point value can be written $m \cdot 2^e$ with $m, e \in \mathbb{Z}$.

An IEEE-754 format imposes

- a precision p: $|m| < 2^p$,
- a minimal exponent $E_{\min}$: $e \geq E_{\min}$,
- a maximal exponent $E_{\max}$: $e \leq E_{\max}$.

When a real number is not representable as a floating-point number, it is rounded in a specific direction.

Example: results are rounded to the nearest floating-point number (tie-breaking to even mantissas).

Floating-point arithmetic

A binary floating-point value can be written $m \cdot 2^e$ with $m, e \in \mathbb{Z}$.

An IEEE-754 format imposes

- a precision p: $|m| < 2^p$,
- a minimal exponent $E_{\min}$: $e \geq E_{\min}$,
- a maximal exponent $E_{\max}$: $e \leq E_{\max}$.

When a real number is not representable as a floating-point number, it is rounded in a specific direction.

Example: results are rounded to the nearest floating-point number (tie-breaking to even mantissas).

Rounding operator: `float<`p`,`$E_{\min}$`,`dir`>`.

Exceptional behaviors

In Gappa's formalism, every data is a real number:

- no signed zeros,
- no Not-a-Numbers,
- no infinities.

Exceptional behaviors

In Gappa's formalism, every data is a real number:

- no signed zeros,
- no Not-a-Numbers,
- no infinities.

Yet, correct behavior of a program can be proved:

- subnormal numbers are handled;

Exceptional behaviors

In Gappa's formalism, every data is a real number:

- no signed zeros,
- no Not-a-Numbers,
- no infinities.

Yet, correct behavior of a program can be proved:

- subnormal numbers are handled;
- the absence of overflow can be expressed and checked:

```
z = float<ieee_32,ne>(x + y);
{ ... -> |z| <= 0x1.FFFFFEp127 }
```

Fixed-point arithmetic

The value of a fixed-point number is $m \cdot 2^E$ with $m \in \mathbb{Z}$.
Rounding operator: `fixed<`E`,`dir`>`.

Fixed-point arithmetic

The value of a fixed-point number is $m \cdot 2^E$ with $m \in \mathbb{Z}$.
Rounding operator: $\texttt{fixed<}E\texttt{,}dir\texttt{>}$.

Theorem on rounding error: ($dir = \texttt{dn}$: rounding toward $-\infty$)

$$\forall x \in \mathbb{R}, \quad \texttt{fixed<}E\texttt{,dn>}(x) - x \in [-2^E, 0].$$

Fixed-point arithmetic

Theorem on rounding error: ($dir = \mathtt{dn}$: rounding toward $-\infty$)

$$\forall x \in \mathbb{R}, \quad \mathtt{fixed<}E\mathtt{,dn>}(x) - x \in [-2^E, 0].$$

Example 1:

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x * y) - x * y in ? }
```

Fixed-point arithmetic

Theorem on rounding error: ($dir = \texttt{dn}$: rounding toward $-\infty$)

$$\forall x \in \mathbb{R}, \quad \texttt{fixed<}E\texttt{,dn>}(x) - x \in [-2^E, 0].$$

Example 1:

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x * y) - x * y in ? }
```

Gappa answers $[-2^{-6}, 0]$. Optimal is $[-0.875 \cdot 2^{-6}, 0]$.

Fixed-point arithmetic

Theorem on rounding error: (dir = `dn`: rounding toward $-\infty$)

$$\forall x \in \mathbb{R}, \quad \texttt{fixed<}E\texttt{,dn>}(x) - x \in [-2^E, 0].$$

Example 1:

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x * y) - x * y in ? }
```

Gappa answers $[-2^{-6}, 0]$. Optimal is $[-0.875 \cdot 2^{-6}, 0]$.

Example 2:

```
{ fixed<-6,dn>(x + y) - (x + y) in ? }
```

Naive answer is $[-2^{-6}, 0]$. Optimal is $[0, 0]$.

Predicate for fixed-point predicate

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x + y) - (x + y) in ? }
```

Optimal interval is $[0, 0]$ because
$x + y = m_x \cdot 2^{-4} + m_y \cdot 2^{-5}$ is representable as a multiple of 2^{-6}.

Predicate for fixed-point predicate

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x + y) - (x + y) in ? }
```

Optimal interval is $[0, 0]$ because
$x + y = m_x \cdot 2^{-4} + m_y \cdot 2^{-5}$ is representable as a multiple of 2^{-6}.

Solution: Gappa internally relies on the predicate

$$\mathsf{FIX}(x, e) \quad \equiv \quad \exists m \in \mathbb{Z}, \quad x = m \cdot 2^e$$

and computes with it:

- $\mathsf{FIX}(x, e_x) \wedge \mathsf{FIX}(y, e_y) \quad \Longrightarrow \quad \mathsf{FIX}(x + y, \min(e_x, e_y))$,
- $\mathsf{FIX}(x, e_x) \wedge \mathsf{FIX}(y, e_y) \quad \Longrightarrow \quad \mathsf{FIX}(x \cdot y, e_x + e_y)$.

Predicate for fixed-point predicate

```
x = fixed<-4,dn>(x_);
y = fixed<-5,dn>(y_);
{ fixed<-6,dn>(x + y) - (x + y) in ? }
```

Optimal interval is $[0,0]$ because
$x + y = m_x \cdot 2^{-4} + m_y \cdot 2^{-5}$ is representable as a multiple of 2^{-6}.

Solution: Gappa internally relies on the predicate

$$\mathrm{FIX}(x, e) \quad \equiv \quad \exists m \in \mathbb{Z}, \quad x = m \cdot 2^e$$

and computes with it:

- $\mathrm{FIX}(x, e_x) \wedge \mathrm{FIX}(y, e_y) \quad \Longrightarrow \quad \mathrm{FIX}(x + y, \min(e_x, e_y))$,
- $\mathrm{FIX}(x, e_x) \wedge \mathrm{FIX}(y, e_y) \quad \Longrightarrow \quad \mathrm{FIX}(x \cdot y, e_x + e_y)$.

Alternate theorem on the rounding error:

$$\forall x \in \mathbb{R}, \quad \mathrm{FIX}(x, E) \quad \Longrightarrow \quad \texttt{fixed<}E\texttt{,dn>}(x) - x = 0.$$

Predicate for floating-point arithmetic

Quantifying the precision required to represent a value:

$$\mathrm{FLT}(x, p) \quad \equiv \quad \exists m, e \in \mathbb{Z}, \quad x = m \cdot 2^e \wedge |m| < 2^p$$

Predicate for floating-point arithmetic

Quantifying the precision required to represent a value:

$$\mathrm{FLT}(x, p) \quad \equiv \quad \exists m, e \in \mathbb{Z}, \quad x = m \cdot 2^e \wedge |m| < 2^p$$

Example: exact floating-point subtraction.

```
@rnd = float< ieee_32, zr >;
a = rnd(a_); b = rnd(b_);
{ a in [3.2,3.3] /\ b in [1.4,1.8] ->
  rnd(a - b) - (a - b) in [0,0] }
```

Note: Sterbenz's lemma does not apply because $\frac{3.3}{1.4} \simeq 2.4$.

Predicate for floating-point arithmetic

Quantifying the precision required to represent a value:

$$\mathsf{FLT}(x, p) \quad \equiv \quad \exists m, e \in \mathbb{Z}, \quad x = m \cdot 2^e \wedge |m| < 2^p$$

Example: exact floating-point subtraction.

```
@rnd = float< ieee_32, zr >;
a = rnd(a_); b = rnd(b_);
{ a in [3.2,3.3] /\ b in [1.4,1.8] ->
  rnd(a - b) - (a - b) in [0,0] }
```

Note: Sterbenz's lemma does not apply because $\frac{3.3}{1.4} \simeq 2.4$.

Gappa automatically proves:

1. $\mathsf{FIX}(a, -22)$ and $\mathsf{FIX}(b, -23)$, hence $\mathsf{FIX}(a - b, -23)$;
2. $|a - b| \leq 1.9$, hence $\mathsf{FLT}(a - b, 24)$.

Hence $a - b = \circ(a - b)$. (Coq proof: 281 lines, 53 proofs)

Outline

1. Introduction
2. Bounding expressions
3. Rounded computations
4. Propagating errors
   - Correlated expressions and interval evaluation
   - Rewriting expressions
   - User-defined rewriting rules
5. Conclusion

Correlated expressions and interval evaluation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that:

- domains of u, v, $\tilde{u}$, and $\tilde{v}$ are $[1, 100]$;
- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \le 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \le 0.2$.

Correlated expressions and interval evaluation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that:

- domains of u, v, $\tilde{u}$, and $\tilde{v}$ are $[1, 100]$;
- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \le 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \le 0.2$.

Interval evaluation:

$$\begin{aligned} \frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} &\in \frac{[1,100]\cdot[1,100]-[1,100]\cdot[1,100]}{[1,100]\cdot[1,100]} \\ &\in \frac{[1,10000]-[1,10000]}{[1,10000]} \\ &\in [-9999, 9999] \end{aligned}$$

Correlated expressions and interval evaluation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that:

- domains of u, v, $\tilde{u}$, and $\tilde{v}$ are $[1, 100]$;
- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \le 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \le 0.2$.

Interval evaluation:

$$\begin{aligned}\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} &\in \frac{[1,100]\cdot[1,100]-[1,100]\cdot[1,100]}{[1,100]\cdot[1,100]}\\ &\in \frac{[1,10000]-[1,10000]}{[1,10000]}\\ &\in [-9999, 9999] \qquad \text{Bad!}\end{aligned}$$

Naive interval arithmetic does not track correlation between values.

Rewriting expressions to reduce decorrelation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that

- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \leq 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \leq 0.2$.

Solution: make correlations explicit.

$$\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} = \frac{\tilde{u}-u}{u} + \frac{\tilde{v}-v}{v} + \frac{\tilde{u}-u}{u}\cdot\frac{\tilde{v}-v}{v}$$

Rewriting expressions to reduce decorrelation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that

- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \le 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \le 0.2$.

Solution: make correlations explicit.

$$\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} = \frac{\tilde{u}-u}{u} + \frac{\tilde{v}-v}{v} + \frac{\tilde{u}-u}{u}\cdot\frac{\tilde{v}-v}{v}$$

Interval evaluation:

$$\begin{aligned}\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} &\in [-0.1, 0.1] + [-0.2, 0.2] + [-0.1, 0.1]\cdot[-0.2, 0.2] \\ &\in [-0.32, 0.32]\end{aligned}$$

Rewriting expressions using similarities

The expression $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ can be seen as the relative error between two similar sub-expressions: $\tilde{u}\cdot\tilde{v}$ and $u\cdot v$.

Rewriting as $\frac{\tilde{u}-u}{u}+\frac{\tilde{v}-v}{v}+\frac{\tilde{u}-u}{u}\cdot\frac{\tilde{v}-v}{v}$ is useful, if tight ranges of $\frac{\tilde{u}-u}{u}$ and $\frac{\tilde{v}-v}{v}$ can be computed.

If $\tilde{u}$ and u are similar expressions, then $\frac{\tilde{u}-u}{u}$ can also be rewritten.

And so on, until Gappa gets to an atomic error, e.g. $\frac{\circ(e)-e}{e}$ that is bounded thanks to a theorem on operator $\circ$.

Rewriting expressions using intermediate terms

Rounding operators prevent these rewritings. E.g. $\frac{\circ(\tilde{u}\cdot\tilde{v})-u\cdot v}{u\cdot v}$.

More generally, how to bound $\frac{\tilde{p}-q}{q}$ when

- $\tilde{p}$ is known to be close to p,
- p is potentially similar to q?

Rewriting expressions using intermediate terms

Rounding operators prevent these rewritings. E.g. $\frac{\circ(\tilde{u} \cdot \tilde{v}) - u \cdot v}{u \cdot v}$.

More generally, how to bound $\frac{\tilde{p} - q}{q}$ when

- $\tilde{p}$ is known to be close to p,
- p is potentially similar to q?

Gappa automatically rewrites the expression so that $\frac{\tilde{p} - p}{p}$ appears:

$$\frac{\tilde{p} - q}{q} = \frac{\tilde{p} - p}{p} + \frac{p - q}{q} + \frac{\tilde{p} - p}{p} \cdot \frac{p - q}{q}$$

Rewriting expressions using intermediate terms

Rounding operators prevent these rewritings. E.g. $\frac{\circ(\tilde{u}\cdot\tilde{v})-u\cdot v}{u\cdot v}$.

More generally, how to bound $\frac{\tilde{p}-q}{q}$ when

- $\tilde{p}$ is known to be close to p,
- p is potentially similar to q?

Gappa automatically rewrites the expression so that $\frac{\tilde{p}-p}{p}$ appears:

$$\frac{\tilde{p}-q}{q} = \frac{\tilde{p}-p}{p} + \frac{p-q}{q} + \frac{\tilde{p}-p}{p} \cdot \frac{p-q}{q}$$

Note:

- $\frac{\tilde{p}-p}{p} = \frac{\circ(\tilde{u}\cdot\tilde{v})-\tilde{u}\cdot\tilde{v}}{\tilde{u}\cdot\tilde{v}}$ is an atomic error;
- $\frac{p-q}{q} = \frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ has similar sub-expressions.

User-defined rewriting rules

When the user is certifying clever code,
automatic rewriting is no longer enough to deal with correlation.

User-defined rewriting rules

When the user is certifying clever code,
automatic rewriting is no longer enough to deal with correlation.

Example: given a fixed-point number $d \in [0.5, 1]$, compute an approximate reciprocal r_2 using two Newton iterations.

▸ Full

```
r1 fixed<-14,dn>=
      r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
  -> r2 - 1/d in ? }
```

User-defined rewriting rules

When the user is certifying clever code,
automatic rewriting is no longer enough to deal with correlation.

Example: given a fixed-point number $d \in [0.5, 1]$, compute an approximate reciprocal r_2 using two Newton iterations.

Full

```
r1 fixed<-14,dn>=
      r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
  -> r2 - 1/d in ? }
```

Gappa's answer: $r_2 - \frac{1}{d} \in [-5.04, 5.05]$.
The tool does not see the correlation between r_2 and $\frac{1}{d}$.

User-defined rewriting rules

Example:

Full

```
r1 fixed<-14,dn>=
      r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
  -> r2 - 1/d in ? }
```

Newton iteration is used for its quadratic convergence.

User-defined rewriting rules

Example: ▸ Full

```
r1 fixed<-14,dn>=
      r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
  -> r2 - 1/d in ? }
```

Newton iteration is used for its quadratic convergence.

So the user just has to tell Gappa about it:

```
r1 ~ r0 * (2 - d * r0);
r0 * (2 - d * r0) - 1/d ->
          (r0 - 1/d) * (r0 - 1/d) * -d;
```

User-defined rewriting rules

Example: Full

```
r1 fixed<-14,dn>=
       r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
   -> r2 - 1/d in ? }
```

Newton iteration is used for its quadratic convergence.

So the user just has to tell Gappa about it:

```
r1 ~ r0 * (2 - d * r0);
r0 * (2 - d * r0) - 1/d ->
            (r0 - 1/d) * (r0 - 1/d) * -d;
```

Answer: $r_2 - \frac{1}{d} \in [-2^{-24.7}, 2^{-28.4}]$. (Coq proof: 774 lines, 138 lemmas)

Outline

Realizations

First public version of Gappa: early 2005.

Since then, around 30 new versions.

Latest version is Gappa 0.7.3.

Realizations

First public version of Gappa: early 2005.
Since then, around 30 new versions.
Latest version is Gappa 0.7.3.

Two separate parts:

1. Gappa tool: 6500 lines of C++;
2. Support library: 8000 lines of Coq proofs.

Realizations

First public version of Gappa: early 2005.
Since then, around 30 new versions.
Latest version is Gappa 0.7.3.

Two separate parts:

1. Gappa tool: 6500 lines of C++;
2. Support library: 8000 lines of Coq proofs.

Perspective: adapt Gappa and write support libraries for other proof checkers: HOL light, PVS, ...

Realizations

It has been successfully used for implementing

- robust floating-point filters in CGAL,
- correctly-rounded elementary functions in CRlibm,
- efficient hardware arithmetic operators,
- etc.

Realizations

It has been successfully used for implementing

- robust floating-point filters in CGAL,
- correctly-rounded elementary functions in CRlibm,
- efficient hardware arithmetic operators,
- etc.

What users say about Gappa:

Higher confidence. Faster development.

Limitations and perspectives of Gappa

Gappa focuses on arithmetic properties of real expressions:

- no arrays of numbers,
- no loops nor conditional execution.

Perspective: interface Gappa with generic certification tools.

Limitations and perspectives of Gappa

Gappa focuses on arithmetic properties of real expressions:

- no arrays of numbers,
- no loops nor conditional execution.

Perspective: interface Gappa with generic certification tools.

Gappa manipulates predicates with numerical parameters:

- no generic error computation for `fixed<n,`*dir*`>` roundings,
- no symbolic domains of expressions.

Perspective: introduce some symbolic computations in Gappa.

Appendix

Intersection and union

```
x = int<dn>(x_);
y = int<dn>(y_);

{ |x| <= 100 /\ |y| <= 100 /\ x * y <= 0
  -> |x + y| <= 100 }

$ x in (-0.5,0.5), y in (-0.5,0.5);
```

Return

User-defined rewriting rules

```
d = fixed<-24,dn>(d_);
r0 = fixed<-8,dn>(r0_);

r1 fixed<-14,dn>=
      r0 * (2 - fixed<-16,dn>(d) * r0);
r2 fixed<-30,dn>= r1 * (2 - d * r1);

{ |r0 - 1/d| <= 1b-8 /\ d in [0.5,1]
  -> r2 - 1/d in ? }

r1 ~ r0 * (2 - d * r0);
r0 * (2 - d * r0) - 1/d ->
          (r0 - 1/d) * (r0 - 1/d) * -d;

r2 ~ r1 * (2 - d * r1);
r1 * (2 - d * r1) - 1/d ->
          (r1 - 1/d) * (r1 - 1/d) * -d;
```

Return

Adding atomic operators to improve interval evaluation

```
{ u in [1,100] /\ v in [1,100] /\
  |(ut - u) / u| <= 0.1 /\
  |(vt - v) / v| <= 0.2
  -> (ut * vt - u * v) / (u * v) in ? }
```

CGAL floating-point filter (partly certified with Gappa)

```
double pqx = qx - px, pqy = qy - py;
double prx = rx - px, pry = ry - py;
double det = pqx * pry - pqy * prx;

double maxx = max(abs(pqx), abs(prx));
double maxy = max(abs(pqy), abs(pry));
double eps = 8.8872057372592758e-16 * maxx * maxy;
if (maxx > maxy) swap(maxx, maxy);

if (maxx < 1e-146) {
  if (maxx == 0) return ZERO;
} else if (maxy < 1e153) {
  if (det > eps)  return POSITIVE;
  if (det < -eps) return NEGATIVE;
}
return UNKNOWN;
```

Rounding to odd and accurate algorithms

Rounding to odd:

$$\square(x) = \begin{cases} x & \text{if } x \text{ is representable by a FP number,} \\ \triangle(x) & \text{if the mantissa of } \triangle(x) \text{ is odd,} \\ \triangledown(x) & \text{otherwise.} \end{cases}$$

Less accurate than rounding to nearest, but satisfy the double-rounding property: $\circ_p(\square_{p+k}(x)) = \circ_p(x)$.

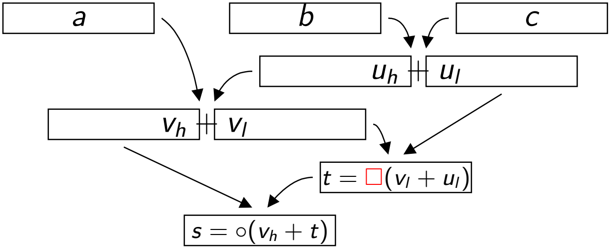
Rounding to odd and accurate algorithms

Rounding to odd:

$$\square(x) = \begin{cases} x & \text{if } x \text{ is representable by a FP number,} \\ \triangle(x) & \text{if the mantissa of } \triangle(x) \text{ is odd,} \\ \triangledown(x) & \text{otherwise.} \end{cases}$$

Less accurate than rounding to nearest, but satisfy the double-rounding property: $\circ_p(\square_{p+k}(x)) = \circ_p(x)$.

Application: correctly-rounded sum s of 3 FP numbers.



Rewriting expressions to reduce decorrelation

Example: compute the range of $\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v}$ knowing that

- values are correlated: $\left|\frac{\tilde{u}-u}{u}\right| \leq 0.1$ and $\left|\frac{\tilde{v}-v}{v}\right| \leq 0.2$.

Solution: make correlations explicit.

$$\frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} = \frac{\tilde{u}-u}{u} + \frac{\tilde{v}-v}{v} + \frac{\tilde{u}-u}{u}\cdot\frac{\tilde{v}-v}{v}$$

Interval evaluation:

$$\begin{aligned} \frac{\tilde{u}\cdot\tilde{v}-u\cdot v}{u\cdot v} &\in [-0.1,0.1]+[-0.2,0.2]+[-0.1,0.1]\cdot[-0.2,0.2] \\ &\in [-0.32,0.32] \end{aligned}$$

But there is still a correlation:

$$\frac{\tilde{u}-u}{u} + \frac{\tilde{v}-v}{v} + \frac{\tilde{u}-u}{u}\cdot\frac{\tilde{v}-v}{v}$$

Adding atomic operators to improve interval evaluation

Goal: a tight range of $p + q + p \cdot q$ with $p \in [\underline{p}, \overline{p}]$ and $q \in [\underline{q}, \overline{q}]$.

Adding atomic operators to improve interval evaluation

Goal: a tight range of $p + q + p \cdot q$ with $p \in [\underline{p}, \overline{p}]$ and $q \in [\underline{q}, \overline{q}]$.

Rewriting: $p + q + p \cdot q = (1 + p) \cdot (1 + q) - 1$.

No correlated expressions anymore, but high precision required.

Adding atomic operators to improve interval evaluation

Goal: a tight range of $p + q + p \cdot q$ with $p \in [\underline{p}, \overline{p}]$ and $q \in [\underline{q}, \overline{q}]$.

Rewriting: $p + q + p \cdot q = (1 + p) \cdot (1 + q) - 1$.

No correlated expressions anymore, but high precision required.

Symbolic interval evaluation assuming $\underline{p} \geq -1$ and $\underline{q} \geq -1$:

$$
\begin{array}{rcl}
p + q + p \cdot q & \in & [1 + \underline{p}, 1 + \overline{p}] \cdot [1 + \underline{q}, 1 + \overline{q}] - [1, 1] \\
& \in & [(1 + \underline{p}) \cdot (1 + \underline{q}) - 1, (1 + \overline{p}) \cdot (1 + \overline{q}) - 1] \\
& \in & [\underline{p} + \underline{q} + \underline{p} \cdot \underline{q}, \overline{p} + \overline{q} + \overline{p} \cdot \overline{q}]
\end{array}
$$

Adding atomic operators to improve interval evaluation

Goal: a tight range of $p + q + p \cdot q$ with $p \in [\underline{p}, \overline{p}]$ and $q \in [\underline{q}, \overline{q}]$.

Rewriting: $p + q + p \cdot q = (1 + p) \cdot (1 + q) - 1$.

No correlated expressions anymore, but high precision required.

Symbolic interval evaluation assuming $\underline{p} \geq -1$ and $\underline{q} \geq -1$:

$$\begin{aligned}
p + q + p \cdot q &\in [1 + \underline{p}, 1 + \overline{p}] \cdot [1 + \underline{q}, 1 + \overline{q}] - [1, 1] \\
&\in [(1 + \underline{p}) \cdot (1 + \underline{q}) - 1, (1 + \overline{p}) \cdot (1 + \overline{q}) - 1] \\
&\in [\underline{p} + \underline{q} + \underline{p} \cdot \underline{q}, \overline{p} + \overline{q} + \overline{p} \cdot \overline{q}]
\end{aligned}$$

Back to the previous example:

$$\frac{\tilde{u} \cdot \tilde{v} - u \cdot v}{u \cdot v} \in [-0.28, 0.32]$$

Bibliography I

- Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion.
  The Boost interval arithmetic library.
  In *Proceedings of the 5th Conference on Real Numbers and Computers*, pages 65–80, Lyon, France, 2003.
- Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion.
  The design of the Boost interval arithmetic library.
  Theoretical Computer Science, 351:111–118, 2006.
- Marc Daumas and Guillaume Melquiond.
  Generating formally certified bounds on values and round-off errors.
  In Vasco Brattka, Christiane Frougny, and Norbert Müller, editors, *Proceedings of the 6th Conference on Real Numbers and Computers*, pages 55–70, Schloß Dagstuhl, Germany, 2004.
- Marc Daumas, Guillaume Melquiond, and César Muñoz.
  Guaranteed proofs using interval arithmetic.
  In Paolo Montuschi and Eric Schwarz, editors, *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, pages 188–195, Cape Cod, MA, USA, 2005.
- Guillaume Melquiond and Sylvain Pion.
  Formally certified floating-point filters for homogeneous geometric predicates.
  Theoretical Informatics and Applications, to be published.

Bibliography II

- Sylvie Boldo, Marc Daumas, William Kahan, and Guillaume Melquiond.
  Proof and certification for an accurate discriminant.
  In *Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, Düsseldorf, Germany, 2006.
- Sylvie Boldo and Guillaume Melquiond.
  When double rounding is odd.
  In *Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics*, Paris, France, 2005.
- Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion.
  Proposing Interval Arithmetic for the C++ Standard.
  In *Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, Düsseldorf, Germany, 2006.
- Florent de Dinechin, Christoph Lauter, and Guillaume Melquiond.
  Assisted verification of elementary functions using Gappa.
  In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 1318–1322, 2006.
- Guillaume Melquiond and Sylvain Pion.
  Formal certification of arithmetic filters for geometric predicates.
  In *Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics*, Paris, France, 2005.

Bibliography III

- Sylvie Boldo and Guillaume Melquiond.
  Emulation of a FMA and correctly-rounded sums: proved algorithms using rounding to odd.
  Research Report HAL inria-00080427, 2006.
  Submitted to *IEEE Transactions on Computers*
- Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion.
  A proposal to add interval arithmetic to the C++ standard library.
  Technical Report 2137, C++ standardization committee, 2006.
- Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion.
  Bool_set: multi-valued logic.
  Technical Report 2136, C++ standardization committee, 2006.