Floating-point arithmetic in the Coq system

Guillaume Melquiond

Mathematical Components INRIA–Microsoft Research

2008-07-08

Verifying an equality, an example

While certifying an algorithm for preventing airplane collision, Carreño and Muñoz had to formally prove:

A plane flying at 250 knots and with a bank angle of 35° has a turn rate of at least 3° each second.

In other words,

$$\frac{3\pi}{180} \leq \frac{\mathsf{g}}{\mathsf{v}} \cdot \mathsf{tan}\left(\frac{35\pi}{180}\right)$$

with $g = 9.8 \ m/s^2$ and $v = 250 \cdot \frac{514}{1000} \ m/s$.

Intro Numbers Basic ops Elementary ops Conclusion

Verifying an equality, an example

The inequality is trivially true since

$$\frac{3\pi}{180} \approx 0.052$$
 and $\frac{g}{v} \cdot \tan\left(\frac{35\pi}{180}\right) \approx 0.053$.

Verifying an equality, an example

The inequality is trivially true since

$$rac{3\pi}{180}pprox 0.052$$
 and $rac{g}{v}\cdot an\left(rac{35\pi}{180}
ight)pprox 0.053.$

Proved in PVS by Lester and Muñoz thanks to interval computations on rational bounds.

Formal proofs and floating-point computations

Kepler's conjecture on 3D sphere packing.

Formal proofs and floating-point computations

Kepler's conjecture on 3D sphere packing.

• Hales splits the problem into thousands of configurations. Each configuration is described by a system with 6 variables, then solved by interval computations on floating-point bounds.

Confidence of the reviewers: only 95% due to the computationally-intensive proof.

Formal proofs and floating-point computations

Kepler's conjecture on 3D sphere packing.

• Hales splits the problem into thousands of configurations. Each configuration is described by a system with 6 variables, then solved by interval computations on floating-point bounds.

Confidence of the reviewers: only 95% due to the computationally-intensive proof.

• Hales now aims to perform a formal proof in order to achieve 100% confidence.

But what about computations?

Computations and proofs

Natural numbers:

```
1 type nat = 0 | S of nat (* 5 = SSSSSO *)
2 let rec plus x y =
3 match x with
4 | 0 -> y
5 | S x' -> plus x' (S y)
```

Computations and proofs

Natural numbers:

1 type nat = 0 | S of nat (* 5 = SSSSSO *)
2 let rec plus x y =
3 match x with
4 | 0 -> y
5 | S x' -> plus x' (S y)

Proof by computation:

1 Goal (plus 3 5) = (plus 5 3). trivial.

Computations and proofs

Natural numbers:

1 type nat = 0 | S of nat (* 5 = SSSSSO *)
2 let rec plus x y =
3 match x with
4 | 0 -> y
5 | S x' -> plus x' (S y)

Proof by computation:

1 Goal (plus 3 5) = (plus 5 3). trivial.

Standard positive integers:

Floating-point arithmetic in the Coq system

Goal: Implement an efficient arithmetic inside a formal system.

- Number representations
- 2 Basic arithmetic operators
- 3 Elementary functions



Outline

Number representations

- FP numbers
- Libraries
- Modules
- Fast integers
- 2 Basic arithmetic operators
- 3 Elementary functions

4 Conclusion

Formats

Definition (Floating-point number)

Floating-point numbers in radix β :

- Pairs of integers $(m, e) \in \mathbb{Z}^2$ interpreted as $m \cdot \beta^e \in \mathbb{R}$.
- Not-a-Number \perp for exceptional behavior: $\frac{1}{0}$, $\sqrt{-1}$, etc.

Formats

Definition (Floating-point number)

Floating-point numbers in radix β :

- Pairs of integers $(m, e) \in \mathbb{Z}^2$ interpreted as $m \cdot \beta^e \in \mathbb{R}$.
- Not-a-Number \perp for exceptional behavior: $\frac{1}{0}$, $\sqrt{-1}$, etc.

Unbounded exponent range \implies no overflow nor underflow. No infinities nor signed zeros.

Rounding

Some real numbers are not representable as floating-point values. E.g. $\frac{1}{3}$ for $\beta = 2$, $\sqrt{2}$ for any β .

Rounding

Some real numbers are not representable as floating-point values. E.g. $\frac{1}{3}$ for $\beta = 2$, $\sqrt{2}$ for any β .

Solution: Use a FP value close to the real number instead. Uniquely defined according to a precision and a rounding direction.

Example: In radix 10 and precision 4, π is rounded to:

- $3141 \cdot 10^{-3}$ when rounding toward $-\infty$ or zero,
- $3142 \cdot 10^{-3}$ when rounding toward $+\infty$ or to nearest.

Rounding

Some real numbers are not representable as floating-point values. E.g. $\frac{1}{3}$ for $\beta = 2$, $\sqrt{2}$ for any β .

Solution: Use a FP value close to the real number instead. Uniquely defined according to a precision and a rounding direction.

Example: In radix 10 and precision 4, π is rounded to:

- $3141 \cdot 10^{-3}$ when rounding toward $-\infty$ or zero,
- $3142 \cdot 10^{-3}$ when rounding toward $+\infty$ or to nearest.

Rounding direction and precision are specified for each operation:

Fadd rnd_UP 500 : $\mathbb{F}_{\beta} \to \mathbb{F}_{\beta} \to \mathbb{F}_{\beta}$.

Existing floating-point libraries

A few libraries on floating-point numbers:

- Moore, Lynch, and Kaufmann's library in ACL2.
- Daumas, Rideau, and Théry's library in Coq.
- Harrison's library in HOL light.

Existing floating-point libraries

A few libraries on floating-point numbers:

- Moore, Lynch, and Kaufmann's library in ACL2.
- Daumas, Rideau, and Théry's library in Coq.
- Harrison's library in HOL light.

Definition of rounding to $-\infty$: downward(format, f, x) = $f \in format \land f \leq x \land \forall g \in format, g \leq x \Rightarrow g \leq f.$

Existing floating-point libraries

A few libraries on floating-point numbers:

- Moore, Lynch, and Kaufmann's library in ACL2.
- Daumas, Rideau, and Théry's library in Coq.
- Harrison's library in HOL light.

Definition of rounding to $-\infty$: downward(format, f, x) = $f \in format \land f \leq x \land \forall g \in format, g \leq x \Rightarrow g \leq f.$

Rounding predicates:

- useful for certifying floating-point programs,
- useless for computing with floating-point numbers.

Modular design

The library provides a generic multi-radix implementation based on Coq's standard integers. But

- an optimized radix-2 arithmetic is better for computations, and/or
- the user may trust the integer arithmetic units of the CPU.

Modular design

The library provides a generic multi-radix implementation based on Coq's standard integers. But

- an optimized radix-2 arithmetic is better for computations, and/or
- the user may trust the integer arithmetic units of the CPU.

Using Coq's module system:

- an interface (module type) describes floating-point operations,
- several implementations (modules) provides them:
 - GenericFloat α : any radix α , standard integers,
 - SpecificFloat StdZRadix2: radix 2
 - SpecificFloat BigIntRadix2:

radix α , standard integers, radix 2, standard integers,

radix 2, fast integers.

Fast integers: Native 31-bit integers

Arithmetic on 31-bit integers has been formally defined in Coq.

- Addition returns an integer and a carry.
- Multiplication returns high and low parts of the result.

Fast integers: Native 31-bit integers

Arithmetic on 31-bit integers has been formally defined in Coq.

- Addition returns an integer and a carry.
- Multiplication returns high and low parts of the result.

Spiwack has modified the virtual machine to improve performance:

- Closed-term integers are compressed into machine words (Ocaml-like encoding).
- Addition, division, rotation, leading-zero count, etc are delegated to the CPU.

Fast integers: Trees of 31-bit integers

Grégoire and Théry have defined an integer type as a binary tree with int31 leaves.



- Logarithmic complexity for accessing digits.
- Divide-and-conquer algorithms, e.g. Karatsuba's multiplication.

 $+/x \div / \sqrt{-}$ Intervals

Addition and multiplication

The set of floating-point numbers $\mathbb{F}_{\beta} - \{\bot\}$ is a ring for the addition and multiplication on real numbers.

FP addition and multiplication first compute the exact value:

•
$$(m_1, e_1) + (m_2, e_2) \rightarrow (m_1 \cdot \beta^{e_1 - e_2} + m_2, e_2)$$
 for $e_2 \le e_1$.
• $(m_1, e_1) \times (m_2, e_2) \rightarrow (m_1 \cdot m_2, e_1 + e_2)$.

This value is then rounded to the target precision.

Division and square root

For the division of (m_1, e_1) by (m_2, e_2) :

- Increase the width of the input: $(m'_1, e'_1) = (m_1 \cdot \beta^k, e_1 k)$, so that $\lfloor m'_1/m_2 \rfloor$ has at least p digits.
- **2** Perform an integer operation on mantissa: $(\lfloor m'_1/m_2 \rfloor, e'_1 e_2)$.
- Round the number to *p* digits, according to the difference $(m'_1 m_2 \cdot \lfloor m'_1/m_2 \rfloor) m_2/2$.

 $+/\times \div/\sqrt{}$ Intervals

Division and square root

For the division of (m_1, e_1) by (m_2, e_2) :

- Increase the width of the input: $(m'_1, e'_1) = (m_1 \cdot \beta^k, e_1 k)$, so that $\lfloor m'_1/m_2 \rfloor$ has at least p digits.
- **2** Perform an integer operation on mantissa: $(\lfloor m'_1/m_2 \rfloor, e'_1 e_2)$.
- Solution Round the number to *p* digits, according to the difference $(m'_1 - m_2 \cdot \lfloor m'_1/m_2 \rfloor) - m_2/2$.

Similarly for the square root of (m, e):

- **2** Perform an integer operation on mantissa: $(\lfloor \sqrt{m'} \rfloor, e'/2)$.
- Solution Round according to $(m' \lfloor \sqrt{m'} \rfloor^2) (2 \cdot \lfloor \sqrt{m'} \rfloor 1)/2$.

Intervals

Definition (Interval)

An interval is a closed connected subset of the real numbers, or \perp_I .

Some subsets can be represented as pairs of FP numbers $\mathbb{I} = \mathbb{F}^2$:

- $x \in [\bot, \bot] \Rightarrow x \in \mathbb{R}$,
- $x \in [\bot, b] \Rightarrow x \in \mathbb{R} \land x \leq b$,
- $x \in [a, \bot] \Rightarrow x \in \mathbb{R} \land a \leq x$,
- $x \in [a, b] \Rightarrow x \in \mathbb{R} \land a \le x \le b$,
- $x \in \perp_I$ holds for any $x \in \mathbb{R} \cup \{\perp\}$.

Interval arithmetic

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

 $\begin{aligned} x \in I_x \land y \in I_y \implies x \diamond y \in I_x \diamond I_y \\ \text{for } \diamond \in \{+, -, \times, \div\}. \text{ Also for unary functions: } \sqrt{}, \ldots \end{aligned}$

Interval arithmetic

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

 $\begin{aligned} x \in I_x \land y \in I_y & \implies \quad x \diamond y \in I_x \diamond I_y \\ \text{for } \diamond \in \{+, -, \times, \div\}. \text{ Also for unary functions: } \sqrt{}, \ldots \end{aligned}$

Arithmetic operations on intervals with FP bounds:

Outline

- Number representations
- 2 Basic arithmetic operators
- 3 Elementary functions
 - Rounding
 - Computation
 - Performances

Conclusion

Elementary functions

Elementary functions: cos, arctan, exp, etc. Difficulty: Their mathematical result *y* cannot generally be split into representable floating-point numbers.

Elementary functions

Elementary functions: cos, arctan, exp, etc. Difficulty: Their mathematical result *y* cannot generally be split into representable floating-point numbers.

No correct rounding for elementary functions:

- They return an interval enclosing the mathematical value.
- Precision is only an hint for the intermediate computations.

Computing elementary functions

Argument reduction until the input is smaller than $\frac{1}{2}$:

$$\begin{cases} \cos x = 2 \cdot (\cos \frac{x}{2})^2 - 1\\ \operatorname{sign}(\sin x) = \operatorname{sign}(\sin \frac{x}{2}) \cdot \operatorname{sign}(\cos \frac{x}{2})\\ \sin x = \operatorname{sign}(\sin x) \cdot \sqrt{1 - (\cos x)^2} \qquad (\text{Only for } |x| \le 2^{20}.) \end{cases}$$

Computing elementary functions

Argument reduction until the input is smaller than $\frac{1}{2}$:

$$\begin{cases} \cos x &= 2 \cdot (\cos \frac{x}{2})^2 - 1\\ \operatorname{sign}(\sin x) &= \operatorname{sign}(\sin \frac{x}{2}) \cdot \operatorname{sign}(\cos \frac{x}{2})\\ \sin x &= \operatorname{sign}(\sin x) \cdot \sqrt{1 - (\cos x)^2} \qquad (\text{Only for } |x| \le 2^{20}.) \end{cases}$$

Evaluation of series until the truncated part is negligible:

$$\begin{aligned} \sin x &= x \cdot \left(1 - \frac{x^2}{6} + \dots + \epsilon\right) & \text{ with } |\epsilon| \leq \frac{x^{2 \cdot n}}{(2 \cdot n + 1)!} \leq \beta^{-p} \\ \text{by interval arithmetic:} \\ \sin x \in X \cdot \left(1 - \frac{X^2}{6} + \dots + \left[\pm \frac{|X|^{2 \cdot n}}{(2 \cdot n + 1)!}\right]\right) & \text{ for } x \in X. \end{aligned}$$

Rough performances

Tests against MPFR data sets (53-bit radix-2 mantissas).

Strategy for correct rounding:

First compute with an intermediate precision p = 63. If the bounds are not rounded to the same 53-bit FP number, try again with an intermediate precision $p \times 1.5$, and so on.

Rough performances

Tests against MPFR data sets (53-bit radix-2 mantissas).

Strategy for correct rounding:

First compute with an intermediate precision p = 63. If the bounds are not rounded to the same 53-bit FP number, try again with an intermediate precision $p \times 1.5$, and so on.

Average delay for evaluating a function: (processor cycles)

Function	Tested values	Coq fast	Coq std	MPFR
arctan	870 (100%)	$20\cdot 10^6$	$10 \cdot 10^{7}$	$57 \cdot 10^{3}$
COS	1289 (91%)	27 · 10 ⁶	$20 \cdot 10^{7}$	$21 \cdot 10^{3}$
sin	1120 (89%)	$32\cdot 10^6$	$81\cdot 10^7$	$21\cdot 10^3$
tan	1297 (81%)	$36\cdot 10^6$	$83 \cdot 10^7$	$22 \cdot 10^3$

Outline

- Number representations
- 2 Basic arithmetic operators
- 3 Elementary functions
- 4 Conclusion
 - Realizations
 - Application

Realizations

Pure library. No modifications to Coq.

11400 lines of Coq:

- floating-point arithmetic: 4600 lines,
- interval arithmetic: 2000 lines,
- tactics:

2000 lines, 2500 lines.

 \Uparrow Automatic solver for inequalities on real-values expressions.

Global positioning requires knowing the local radius of Earth:

$$r_p(\phi) = \frac{a}{\sqrt{1 + (1 - f)^2 \cdot \tan^2 \phi}}$$

This can be approximated by a degree-5 polynomial P with single-precision coefficients: $P(\phi_m^2 - \phi^2) = \tilde{r_p}(\phi)$.

$$P(x) = \frac{4439091}{4} + x \cdot \left(\frac{9023647}{4} + x \cdot \left(\dots \frac{6661427}{131072}\right)\right)$$

Goal: Prove $\left|\frac{r_{\rho}(\phi) - \tilde{r_{\rho}}(\phi)}{r_{\rho}(\phi)}\right| \le 23 \cdot 2^{-24}$ when $0 \le \phi \le \phi_m = \frac{715}{512}$.

Inequality proved with order-1 Taylor interval approximations on $\sim 10^5$ sub-intervals of $[0,\phi_m].$

Inequality proved with order-1 Taylor interval approximations on $\sim 10^5$ sub-intervals of $[0,\phi_m].$

 Daumas, Melquiond, and Muñoz, in PVS (2005).
 An oracle chooses the best sub-intervals and it generates one PVS script per sub-interval.
 Verification: several hours on a 48-core parallel computer.

Inequality proved with order-1 Taylor interval approximations on $\sim 10^5$ sub-intervals of $[0,\phi_m].$

- Daumas, Melquiond, and Muñoz, in PVS (2005).
 An oracle chooses the best sub-intervals and it generates one PVS script per sub-interval.
 Verification: several hours on a 48-core parallel computer.
- In Coq (2008), a few minutes on a laptop computer.

Inequality proved with order-1 Taylor interval approximations on $\sim 10^5$ sub-intervals of $[0,\phi_m].$

- Daumas, Melquiond, and Muñoz, in PVS (2005).
 An oracle chooses the best sub-intervals and it generates one PVS script per sub-interval.
 Verification: several hours on a 48-core parallel computer.
- In Coq (2008), a few minutes on a laptop computer.
 Differences: No oracle, but floating-point numbers.

Intro Numbers Basic ops Elementary ops Conclusion

Realizations Application

Verifying inequalities, another example

```
Require Import Reals.
  Require Import tactics.
 3 Open Local Scope R_scope.
 4
 5
   Definition a := 6378137.
 6 Definition f := 1000000000/298257223563.
   Definition umf2 := (1 - f)^2.
 8 Definition max := 715/512.
 9 Definition rp phi := a / sqrt (1 + umf2 * (tan phi)^2).
10 Definition arp phi :=
     let x := max^{2} - phi<sup>2</sup> in
11
12 4439091/4 + x * (
13
        9023647/4 + x * (
          13868737/64 + x * (
14
15
            13233647/2048 + x * (
16
              -1898597/16384 + x *
17
                 (-6661427/131072)))).
18
19 Goal
20
     forall phi, 0 <= phi <= max ->
21
     Rabs ((rp phi - arp phi) / rp phi) <= 23/16777216.
22 Proof.
23
     unfold rp, arp, umf2, a, f, max. intros.
24
     Time interval with (i_bisect_diff phi, i_nocheck).
                                                            (* 4s *)
(* 96s *)
25 Time Qed.
```

Tactic parameters:

bisection and order-1 evaluation on ϕ , check delayed at Qed time, minimal relative width of intervals: 2^{-15} , precision: 30 bits.

Realizations Application

Questions?

Mail: guillaume.melquiond@inria.fr Web: http://www.msr-inria.inria.fr/soft/coq-interval/