# Floating-point arithmetic in the Coq system 

Guillaume Melquiond<br>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 $35^{\circ}$ has a turn rate of at least $3^{\circ}$ each second.

In other words,

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

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

## Verifying an equality, an example

The inequality is trivially true since

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

## Verifying an equality, an example

The inequality is trivially true since

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

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 \(\quad(* 5=\) SSSSSO *)
2 let rec plus x y =
3 match x with
4 | O -> y
5 | S x' -> plus x' (S y)
```


## Computations and proofs

Natural numbers:

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

$$
\begin{aligned}
& 1 \text { type nat }=0 \text { | S of nat } \quad(* 5=\text { SSSSSO *) } \\
& 2 \text { let rec plus x y = } \\
& 3 \text { match x with } \\
& 4 \text { | } 0 \text {-> y } \\
& 5 \text { | S x' -> plus x' (S y) }
\end{aligned}
$$

Proof by computation:

$$
1 \text { Goal (plus 3 5) = (plus 5 3). trivial. }
$$

Standard positive integers:

$$
\begin{aligned}
& 1 \text { type pos }=\mathrm{H} \text { I B0 of pos I B1 of pos } \\
& 2 \\
& 2(* 26=\text { B0B1B0B1H } *)
\end{aligned}
$$

## Floating-point arithmetic in the Coq system

Goal: Implement an efficient arithmetic inside a formal system.
(1) Number representations
(2) Basic arithmetic operators
(3) Elementary functions
4) Conclusion

## Outline

(1) 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 $\beta$ :

- 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 $\beta$ :

- 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 $\Longrightarrow$ 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 $\beta$.

## Rounding

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

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 $4, \pi$ 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 $\beta$.

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 $4, \pi$ 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:

$$
\text { Fadd rnd_UP } 500: \mathbb{F}_{\beta} \rightarrow \mathbb{F}_{\beta} \rightarrow \mathbb{F}_{\beta}
$$

## 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 $\wedge f \leq x \wedge \forall g \in$ 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 $\wedge f \leq x \wedge \forall g \in$ 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 $\alpha$ :
- SpecificFloat StdZRadix2:
- SpecificFloat BigIntRadix2:
any radix $\alpha$, 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.


## Addition and multiplication

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

FP addition and multiplication first compute the exact value:

- $\left(m_{1}, e_{1}\right)+\left(m_{2}, e_{2}\right) \rightarrow\left(m_{1} \cdot \beta^{e_{1}-e_{2}}+m_{2}, e_{2}\right)$ for $e_{2} \leq e_{1}$.
- $\left(m_{1}, e_{1}\right) \times\left(m_{2}, e_{2}\right) \rightarrow\left(m_{1} \cdot m_{2}, e_{1}+e_{2}\right)$.

This value is then rounded to the target precision.

## Division and square root

For the division of $\left(m_{1}, e_{1}\right)$ by $\left(m_{2}, e_{2}\right)$ :
(1) Increase the width of the input: $\left(m_{1}^{\prime}, e_{1}^{\prime}\right)=\left(m_{1} \cdot \beta^{k}, e_{1}-k\right)$, so that $\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor$ has at least $p$ digits.
(2) Perform an integer operation on mantissa: $\left(\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor, e_{1}^{\prime}-e_{2}\right)$.
(3) Round the number to $p$ digits, according to the difference $\left(m_{1}^{\prime}-m_{2} \cdot\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor\right)-m_{2} / 2$.

## Division and square root

For the division of $\left(m_{1}, e_{1}\right)$ by $\left(m_{2}, e_{2}\right)$ :
(1) Increase the width of the input: $\left(m_{1}^{\prime}, e_{1}^{\prime}\right)=\left(m_{1} \cdot \beta^{k}, e_{1}-k\right)$, so that $\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor$ has at least $p$ digits.
(2) Perform an integer operation on mantissa: $\left(\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor, e_{1}^{\prime}-e_{2}\right)$.
(3) Round the number to $p$ digits, according to the difference $\left(m_{1}^{\prime}-m_{2} \cdot\left\lfloor m_{1}^{\prime} / m_{2}\right\rfloor\right)-m_{2} / 2$.

Similarly for the square root of $(m, e)$ :
(2) Perform an integer operation on mantissa: $\left(\left\lfloor\sqrt{m^{\prime}}\right\rfloor, e^{\prime} / 2\right)$.
(3) Round according to $\left(m^{\prime}-\left\lfloor\sqrt{m^{\prime}}\right\rfloor^{2}\right)-\left(2 \cdot\left\lfloor\sqrt{m^{\prime}}\right\rfloor-1\right) / 2$.

## Intervals

## Definition (Interval)

An interval is a closed connected subset of the real numbers, or $\perp_{/}$.

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

- $x \in[\perp, \perp] \Rightarrow x \in \mathbb{R}$,
- $x \in[\perp, b] \Rightarrow x \in \mathbb{R} \wedge x \leq b$,
- $x \in[a, \perp] \Rightarrow x \in \mathbb{R} \wedge a \leq x$,
- $x \in[a, b] \Rightarrow x \in \mathbb{R} \wedge a \leq x \leq b$,
- $x \in \perp_{\text {I }}$ 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:

$$
x \in I_{x} \wedge y \in I_{y} \quad \Longrightarrow \quad x \diamond y \in I_{x} \diamond I_{y}
$$

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

## Interval arithmetic

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

$$
x \in I_{x} \wedge y \in I_{y} \quad \Longrightarrow \quad x \diamond y \in I_{x} \diamond I_{y}
$$

for $\diamond \in\{+,-, \times, \div\}$. Also for unary functions: $\sqrt{ }, \ldots$
Arithmetic operations on intervals with FP bounds:

- $[a, b]+[c, d]=[\nabla(a+c), \triangle(b+d)]$,
- $[a, b]-[c, d]=[\nabla(a-d), \triangle(b-c)]$,
- . . .


## Outline

## (1) Number representations

(2) Basic arithmetic operators
(3) Elementary functions

- Rounding
- Computation
- Performances

4 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 $\frac{1}{2}$ :

$$
\begin{aligned}
& \left\{\begin{aligned}
\cos x & =2 \cdot\left(\cos \frac{x}{2}\right)^{2}-1 \\
\operatorname{sign}(\sin x) & =\operatorname{sign}\left(\sin \frac{x}{2}\right) \cdot \operatorname{sign}\left(\cos \frac{x}{2}\right)
\end{aligned}\right. \\
& \sin x=\operatorname{sign}(\sin x) \cdot \sqrt{1-(\cos x)^{2}} \quad\left(\text { Only for }|x| \leq 2^{20} .\right)
\end{aligned}
$$

## Computing elementary functions

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

$$
\left\{\begin{aligned}
\cos x & =2 \cdot\left(\cos \frac{x}{2}\right)^{2}-1 \\
\operatorname{sign}(\sin x) & =\operatorname{sign}\left(\sin \frac{x}{2}\right) \cdot \operatorname{sign}\left(\cos \frac{x}{2}\right)
\end{aligned}\right.
$$

$$
\sin x=\operatorname{sign}(\sin x) \cdot \sqrt{1-(\cos x)^{2}} \quad\left(\text { Only for }|x| \leq 2^{20} .\right)
$$

Evaluation of series until the truncated part is negligible:

$$
\sin x=x \cdot\left(1-\frac{x^{2}}{6}+\cdots+\epsilon\right) \quad \text { with }|\epsilon| \leq \frac{x^{2 \cdot n}}{(2 \cdot n+1)!} \leq \beta^{-p}
$$

by interval arithmetic:
$\sin x \in X \cdot\left(1-\frac{X^{2}}{6}+\cdots+\left[ \pm \frac{|X|^{2 \cdot n}}{(2 \cdot n+1)!}\right]\right)$ for $x \in X$.

## 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 |
| :--- | ---: | :---: | :---: | :---: |
| $\arctan$ | $870(100 \%)$ | $20 \cdot 10^{6}$ | $10 \cdot 10^{7}$ | $57 \cdot 10^{3}$ |
| $\cos$ | $1289(91 \%)$ | $27 \cdot 10^{6}$ | $20 \cdot 10^{7}$ | $21 \cdot 10^{3}$ |
| $\sin$ | $1120(89 \%)$ | $32 \cdot 10^{6}$ | $81 \cdot 10^{7}$ | $21 \cdot 10^{3}$ |
| $\tan$ | $1297(81 \%)$ | $36 \cdot 10^{6}$ | $83 \cdot 10^{7}$ | $22 \cdot 10^{3}$ |

## Outline

## (1) 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: 2500 lines.
$\Uparrow$ Automatic solver for inequalities on real-values expressions.


## Verifying inequalities, another example

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: $P\left(\phi_{m}^{2}-\phi^{2}\right)=\tilde{r_{p}}(\phi)$.

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

Goal: Prove $\left|\frac{r_{p}(\phi)-r_{p}(\phi)}{r_{p}(\phi)}\right| \leq 23 \cdot 2^{-24}$ when $0 \leq \phi \leq \phi_{m}=\frac{715}{512}$.

## Verifying inequalities, another example

Inequality proved with order-1 Taylor interval approximations on $\sim 10^{5}$ sub-intervals of $\left[0, \phi_{m}\right]$.

## Verifying inequalities, another example

Inequality proved with order-1 Taylor interval approximations on $\sim 10^{5}$ sub-intervals of $\left[0, \phi_{m}\right]$.
(1) 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.

## Verifying inequalities, another example

Inequality proved with order-1 Taylor interval approximations on $\sim 10^{5}$ sub-intervals of $\left[0, \phi_{m}\right]$.
(1) 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.
(2) In Coq (2008), a few minutes on a laptop computer.

## Verifying inequalities, another example

Inequality proved with order-1 Taylor interval approximations on $\sim 10^{5}$ sub-intervals of $\left[0, \phi_{m}\right]$.
(1) 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.
(2) In Coq (2008), a few minutes on a laptop computer. Differences: No oracle, but floating-point numbers.

## Verifying inequalities, another example

```
Require Import Reals.
Require Import tactics.
Open Local Scope R_scope.
Definition a := 6378137.
Definition f := 1000000000/298257223563.
Definition umf2 := (1 - f)}\mp@subsup{}{}{2}
Definition max := 715/512.
Definition rp phi := a / sqrt (1 + umf2 * (tan phi)}\mp@subsup{}{2}{2})
Definition arp phi :=
    let x := max}\mp@subsup{}{}{2}-\mp@subsup{p}{i}{2}\mathrm{ in
    4439091/4 + x * (
        9023647/4 + x * (
            13868737/64 + x * 
                13233647/2048 + x * (
                -1898597/16384 + x *
                    (-6661427/131072))))).
Goal
    forall phi, 0 <= phi <= max ->
    Rabs ((rp phi - arp phi) / rp phi) <= 23/16777216.
Proof.
    unfold rp, arp, umf2, a, f, max. intros.
```



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

## Questions?

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

