

# Certified exact real arithmetic using co-induction in arbitrary integer radix

Nicolas Julien

INRIA Sophia-Antipolis, Marelle project

Everest/Marelle seminar

# Outline

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms
- 4 Formalisation and certification
- 5 Results
- 6 Perspectives

# Plan

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms
- 4 Formalisation and certification
- 5 Results
- 6 Perspectives

# Usual representation of real numbers

Flotting numbers (IEEE 754)

- ▶ Effective computation
- ▶ Low memory use

# Usual representation of real numbers

Flotting numbers (IEEE 754)

- ▶ Effective computation
- ▶ Low memory use

But

- ▶ Actually a finite subset of  $\mathbb{Q}$
- ▶ Roundness problem
- ▶ Lack of usual real numbers properties

# Motivations

Exact real arithmetic

- ▶ Arbitrary precision computation

Formal proof

- ▶ Computation correctness
- ▶ Use computation as a proof

Arbitrary integer radix

- ▶ Use microprocessor integers

# Plan

- 1 Motivations
- 2 Representation**
- 3 Examples of algorithms
- 4 Formalisation and certification
- 5 Results
- 6 Perspectives

## Representation used

- ▶ A real number  $r$  of  $[-1, 1]$  in radix  $\beta$
- ▶ Infinite stream  $s$  of **signed digits** of  $\beta$

## Representation used

- ▶ A real number  $r$  of  $[-1, 1]$  in radix  $\beta$
- ▶ Infinite stream  $s$  of **signed digits** of  $\beta$
- ▶  $[s]_{\beta} = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i}$
- ▶ with  $-\beta < d_i < \beta$

# Representation used

- ▶ A real number  $r$  of  $[-1, 1]$  in radix  $\beta$
- ▶ Infinite stream  $s$  of signed digits of  $\beta$
- ▶ 
$$[s]_{\beta} = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i}$$
- ▶ with  $-\beta < d_i < \beta$
- ▶ In radix 10,  $\frac{1}{3} : 33333333 \dots$

## Representation used

- ▶ A real number  $r$  of  $[-1, 1]$  in radix  $\beta$
- ▶ Infinite stream  $s$  of signed digits of  $\beta$
- ▶ 
$$[s]_{\beta} = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i}$$
- ▶ with  $-\beta < d_i < \beta$
- ▶ In radix 10,  $\frac{1}{3} : 33333333 \dots$
- ▶ Notice  $[d1 :: s]_{\beta} = \frac{d1 + [s]_{\beta}}{\beta}$

## Representation used

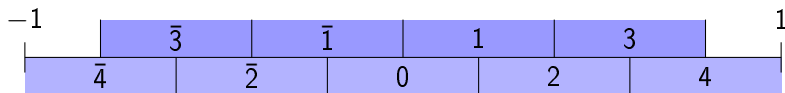
- ▶ A real number  $r$  of  $[-1, 1]$  in radix  $\beta$
- ▶ Infinite stream  $s$  of **signed digits** of  $\beta$
- ▶ 
$$[s]_{\beta} = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i}$$
- ▶ with  $-\beta < d_i < \beta$
- ▶ In radix 10,  $\frac{1}{3} : 33333333 \dots$
- ▶ Notice  $[d1 :: s]_{\beta} = \frac{d1 + [s]_{\beta}}{\beta}$
- ▶ We add an exponent to have all real numbers
- ▶  $[(s, e)]_{\beta} = \beta^e [s]_{\beta}$

# Why this representation ?

- ▶ Close to computer
- ▶ No need for a lot of mathematics

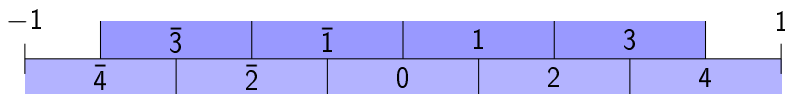
# Why this representation ?

- ▶ Close to computer
- ▶ No need for a lot of mathematics
- ▶ Redundant representation



## Why this representation ?

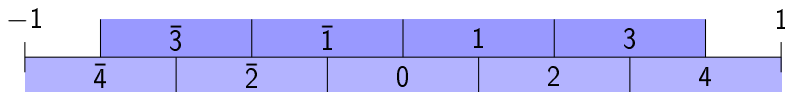
- ▶ Close to computer
- ▶ No need for a lot of mathematics
- ▶ Redundant representation



- ▶ Knowing the result with a precision of  $\frac{1}{2\beta} \Rightarrow$  enough to find a possible first digit
- ▶ Digits are produced 1 by 1

## Why this representation ?

- ▶ Close to computer
- ▶ No need for a lot of mathematics
- ▶ Redundant representation



- ▶ Knowing the result with a precision of  $\frac{1}{2\beta} \Rightarrow$  enough to find a possible first digit
- ▶ Digits are produced 1 by 1
- ▶ But comparison is not decidable

# Plan

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms**
- 4 Formalisation and certification
- 5 Results
- 6 Perspectives

# Computation of addition

- ▶ Result of addition in  $[-2, 2]$  : cannot always be represented

# Computation of addition

- ▶ Result of addition in  $[-2, 2]$  : cannot always be represented
- ▶ Computation of  $\frac{x + y + r}{\beta}$ ,  $r \in [-\beta + 2, \beta - 2]$

# Computation of addition

- ▶ Result of addition in  $[-2, 2]$  : cannot always be represented
- ▶ Computation of  $\frac{x + y + r}{\beta}$ ,  $r \in [-\beta + 2, \beta - 2]$
- ▶ We compute the first digit of  $x$  and  $y$

$$\frac{\frac{x_1 + x'}{\beta} + \frac{y_1 + y'}{\beta} + r}{\beta} = \frac{\frac{x_1 + y_1}{\beta} + r + \frac{x' + y'}{\beta}}{\beta}$$

# Computation of addition

- ▶ Result of addition in  $[-2, 2]$  : cannot always be represented
- ▶ Computation of  $\frac{x + y + r}{\beta}$ ,  $r \in [-\beta + 2, \beta - 2]$
- ▶ We compute the first digit of  $x$  and  $y$

$$\frac{\frac{x_1 + x'}{\beta} + \frac{y_1 + y'}{\beta} + r}{\beta} = \frac{\frac{x_1 + y_1}{\beta} + r + \frac{x' + y'}{\beta}}{\beta}$$

- ▶ We compute  $q \in \{-1, 0, 1\}$ ,  $r' \in \{-\beta + 2, \dots, \beta + 2\}$   

$$x_1 + x_2 = q \times \beta + r'$$

# Computation of addition

- ▶ Result of addition in  $[-2, 2]$  : cannot always be represented
- ▶ Computation of  $\frac{x + y + r}{\beta}$ ,  $r \in [-\beta + 2, \beta - 2]$
- ▶ We compute the first digit of  $x$  and  $y$

$$\frac{\frac{x_1 + x'}{\beta} + \frac{y_1 + y'}{\beta} + r}{\beta} = \frac{\frac{x_1 + y_1}{\beta} + r + \frac{x' + y'}{\beta}}{\beta}$$

- ▶ We compute  $q \in \{-1, 0, 1\}$ ,  $r' \in \{-\beta + 2, \dots, \beta + 2\}$

$$x_1 + x_2 = q \times \beta + r'$$

- ▶ We can return  $(q + r) :: \frac{x' + y' + r'}{\beta}$

# Computation of series

- ▶ Idea : divide the series in one representative part and one close to zero

$$\sum_{i=0}^{\infty} a_i = \sum_{i=0}^n a_i + \sum_{i=n+1}^{\infty} a_i, \quad \left| \sum_{i=n+1}^{\infty} a_i \right| \leq \frac{\beta - 2}{2\beta^2}$$

# Computation of series

- ▶ Idea : divide the series in one representative part and one close to zero

$$\sum_{i=0}^{\infty} a_i = \sum_{i=0}^n a_i + \sum_{i=n+1}^{\infty} a_i, \quad \left| \sum_{i=n+1}^{\infty} a_i \right| \leq \frac{\beta - 2}{2\beta^2}$$

- ▶ Computing the first 2 digits of  $\sum_{i=0}^n a_i$  gives us to an approximation of magnitude  $\frac{1}{2\beta}$  of the series

# Computation of series

- ▶ Idea : divide the series in one representative part and one close to zero

$$\sum_{i=0}^{\infty} a_i = \sum_{i=0}^n a_i + \sum_{i=n+1}^{\infty} a_i, \quad \left| \sum_{i=n+1}^{\infty} a_i \right| \leq \frac{\beta - 2}{2\beta^2}$$

- ▶ Computing the first 2 digits of  $\sum_{i=0}^n a_i$  gives us to an approximation of magnitude  $\frac{1}{2\beta}$  of the series
- ▶ Thus we can compute the first digit
- ▶ This computation does not depend on the series
- ▶ It is described in the function `make_digit`

## Example described series

- ▶ Multiplication :  $x \times y = \sum_{i=1}^{\infty} \frac{x_i \times y}{\beta^i}$
- ▶ Euler constant:  $e - 2 = \sum_{i=2}^{\infty} \frac{1}{i!}$
- ▶  $\frac{\pi}{4} = \arctan \frac{1}{2} + \arctan \frac{1}{3}$ ,  $\arctan \frac{1}{n} = \sum_{i=0}^{\infty} \frac{(-1)^i \frac{1}{n^{2i+1}}}{2i+1}$  (ln radix 2)

# Plan

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms
- 4 Formalisation and certification**
- 5 Results
- 6 Perspectives

# Co-induction in Coq

- ▶ Definition of infinite object types

```
CoInductive stream (A: Set) :=  
  Cons : A → stream A → stream A.
```

# Co-induction in Coq

- ▶ Definition of infinite object types

```
CoInductive stream (A: Set) :=  
  Cons : A → stream A → stream A.
```

- ▶ Co-recursive function to build these infinite objects
  - ▶ Guard condition to prevent infinite loops

```
CoFixpoint zero : stream ℤ := Cons 0 zero.
```

## Co-induction in Coq

- ▶ Definition of infinite object types

```
CoInductive stream (A: Set) :=
  Cons : A → stream A → stream A.
```

- ▶ Co-recursive function to build these infinite objects
  - ▶ Guard condition to prevent infinite loops

```
CoFixpoint zero : stream ℤ := Cons 0 zero.
```

- ▶ Co-inductive predicates to describe properties of infinite objects

```
CoInductive stream_digit_pos : Stream ℤ → Prop
  I : ∀ d s, 0 ≤ d → stream_digit_pos s →
    stream_digit_pos (Cons d s).
```

- ▶ Infinite proof : proof term is a co-inductive object
- ▶ Tactic `cofix`

# Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers

# Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers
  - ▶ Show properties of these functions that characterize real numbers

# Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers
  - ▶ Show properties of these functions that characterize real numbers
  - ▶ Bind our representation to an existing definition

# Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers
  - ▶ Show properties of these functions that characterize real numbers
  - ▶ Bind our representation to an existing definition
- ▶ We used the axiomatized standard definition of reals in Coq

# Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers
  - ▶ Show properties of these functions that characterize real numbers
  - ▶ Bind our representation to an existing definition
- ▶ We used the axiomatized standard definition of reals in Coq
  - ▶ If the stream  $s$  represents the real number  $r$  of  $[-1, 1]$
  - ▶ And if  $k$  is a digit of  $\beta$
  - ▶ Then stream  $k :: s$  represents  $\frac{k+r}{\beta}$

## Certification of computations

- ▶ How to certify our algorithms properly describe the functions on real numbers
  - ▶ Show properties of these functions that characterize real numbers
  - ▶ Bind our representation to an existing definition
- ▶ We used the axiomatized standard definition of reals in Coq
  - ▶ If the stream  $s$  represents the real number  $r$  of  $[-1, 1]$
  - ▶ And if  $k$  is a digit of  $\beta$
  - ▶ Then stream  $k :: s$  represents  $\frac{k+r}{\beta}$

**CoInductive** represents (b :  $\mathbb{Z}$ ): stream  $\mathbb{Z}$   $\rightarrow$   $\mathbb{R}$   $\rightarrow$  **Prop** :=  
 | rep :  $\forall$  (s : stream  $\mathbb{Z}$ ) (r :  $\mathbb{R}$ ) (k :  $\mathbb{Z}$ ),  
 represents b s r  $\rightarrow$   
 -1  $\leq$  r  $\leq$  1  $\rightarrow$   
 -b < k < b  $\rightarrow$   
 represents b (k :: s)  $\frac{k+r}{b}$ .

# Correctness proof

- ▶ Link the result of an algorithm  $F$  to the result of the fonction  $f$  we

hope the algorithm computes

$$\begin{array}{ccc}
 s_1, \dots, s_n & \rightarrow & F(s_1, \dots, s_n) \\
 \downarrow & & \downarrow \\
 r_1, \dots, r_n & \rightarrow & f(r_1, \dots, r_n)
 \end{array}$$

- ▶ Exemple of addition

**Theorem** `add_str_correct` :

$\forall (b\ r : \mathbb{Z}) (x\ y : \text{stream } \mathbb{Z}) (u\ v : \mathbb{R}),$

represents  $b\ x\ u \rightarrow$

represents  $b\ y\ v \rightarrow$

$-b + 2 \leq r \leq b - 2 \rightarrow$

represents  $b\ (\text{add\_str } b\ x\ y\ r) \frac{u + v + r}{b}.$

# Proof difficulties

- ▶ Guard condition of co-induction
  - ▶ Automation is possible
- ▶ Writing series and their proofs
  - ▶ Simplified by function `make_digit` and its certification
- ▶ Formalization of radix  $\Rightarrow$  non linear inequalities
- ▶ Modular arithmetic

# Plan

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms
- 4 Formalisation and certification
- 5 Results**
- 6 Perspectives

# Results (Theory)

- ▶ Computation of the result of a function  $f$  with a precision  $\frac{1}{n}$

$$\frac{\lceil \log n \rceil \times T(f)}{\log \beta}$$

- ▶  $\frac{\lceil \log n \rceil}{\log \beta}$  digits to compute to have a precision  $\frac{1}{n}$
- ▶  $T(f)$  : Time to compute one digit in the function  $f$

## Results

- ▶ Computation of  $\frac{1}{3} + \frac{1}{7}$

Used radix	LCR	$2^2$	$2^8$	$2^{16}$	$2^{32}$
Number of computed digits	6400	3200	800	400	200
Time (s)	31	6.184	0.440	0.284	0.308

## Results

- Computation of  $\frac{1}{3} + \frac{1}{7}$

Used radix	LCR	$2^2$	$2^8$	$2^{16}$	$2^{32}$
Number of computed digits	6400	3200	800	400	200
Time (s)	31	6.184	0.440	0.284	0.308

- Computation  $\frac{1}{3} \times \frac{1}{7}$

Used radix	LCR	$2^2$	$2^8$	$2^{16}$	$2^{32}$
Number of computed digits	640	320	80	40	20
Time (s)	0.028	16.96	1.412	1.224	1.768

## Results

- ▶ Computation of  $\frac{1}{3} + \frac{1}{7}$

Used radix	LCR	$2^2$	$2^8$	$2^{16}$	$2^{32}$
Number of computed digits	6400	3200	800	400	200
Time (s)	31	6.184	0.440	0.284	0.308

- ▶ Computation  $\frac{1}{3} \times \frac{1}{7}$

Used radix	LCR	$2^2$	$2^8$	$2^{16}$	$2^{32}$
Number of computed digits	640	320	80	40	20
Time (s)	0.028	16.96	1.412	1.224	1.768

- ▶ Computation of  $e - 2$

Base utilisée	LCR	10	32	64	100	$2^{10}$
Number of computed digits	300	90	60	50	45	30
Time (s)	2.3	19.88	13.60	1.516	3.420	604

# Plan

- 1 Motivations
- 2 Representation
- 3 Examples of algorithms
- 4 Formalisation and certification
- 5 Results
- 6 Perspectives**

# Perspectives

- ▶ Improvement of multiplication (almost done ...)
- ▶ Formalization of digits and big numbers type (almost done ...)
- ▶ Computation of formal series
- ▶ Computation of inverse :  $\frac{1}{x} = \frac{1}{1-(1-x)} = \sum_{i=0}^{\infty} (1-x)^i$
- ▶ Computation of analytic functions