Effective Analysis: Foundations, Implementations, Certification
Luminy, France

Formal verification of numerical analysis programs

Sylvie Boldo

Inria

January 12th, 2016

Effective Analysis:

Foundations, Implementations, Certification

Effective Analysis:

Foundations, Implementations, Certification

(Applied) Mathematics

Introduction

Effective Analysis:

Foundations, Implementations, Certification

(Applied) Mathematics

and

Formal proof (Coq)

Introduction

Effective Analysis:

Foundations, Implementations, Certification

(Applied) Mathematics

and

Formal proof (Coq)

and

`Floating-point numbers`

Motivations

PDE (Partial Differential Equation) $\Rightarrow$ weather forecast

$\Rightarrow$ nuclear simulation

$\Rightarrow$ optimal control

$\Rightarrow$ ...

Motivations

PDE (Partial Differential Equation) $\Rightarrow$ weather forecast
$\Rightarrow$ nuclear simulation
$\Rightarrow$ optimal control
$\Rightarrow$ ...

Usually too complex to solve by an exact mathematical formula
$\Rightarrow$ approximated by numerical scheme over discrete grids

$\Rightarrow$ mathematical proofs of the convergence of the numerical scheme
(we compute something close to the PDE solution if the grids size decreases)

Motivations

PDE (Partial Differential Equation) $\Rightarrow$ weather forecast
$\Rightarrow$ nuclear simulation
$\Rightarrow$ optimal control
$\Rightarrow$ ...

Usually too complex to solve by an exact mathematical formula
$\Rightarrow$ approximated by numerical scheme over discrete grids

$\Rightarrow$ mathematical proofs of the convergence of the numerical scheme (we compute something close to the PDE solution if the grids size decreases)

$\Rightarrow$ C program implementing the scheme

Motivations

PDE (Partial Differential Equation) $\Rightarrow$ weather forecast
$\Rightarrow$ nuclear simulation
$\Rightarrow$ optimal control
$\Rightarrow$ ...

Usually too complex to solve by an exact mathematical formula
$\Rightarrow$ approximated by numerical scheme over discrete grids

$\Rightarrow$ mathematical proofs of the convergence of the numerical scheme (we compute something close to the PDE solution if the grids size decreases)

$\Rightarrow$ C program implementing the scheme

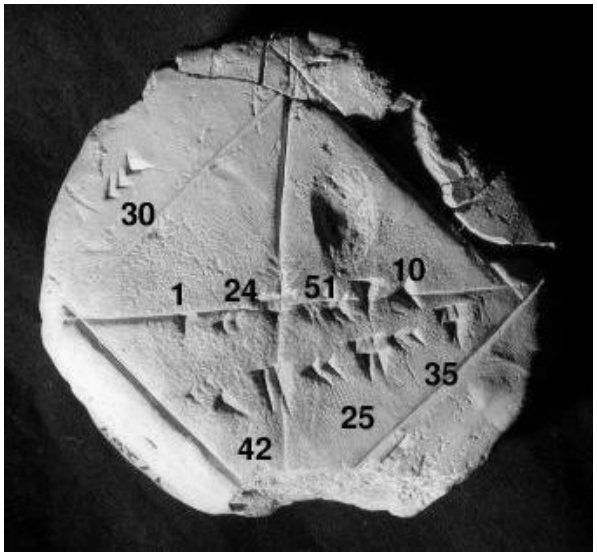
Let us machine-check this kind of programs!

Joint work with

- François Clément
- Jean-Christophe Filliâtre
- Vincent Martin
- Micaela Mayero
- Guillaume Melquiond
- Pierre Weis

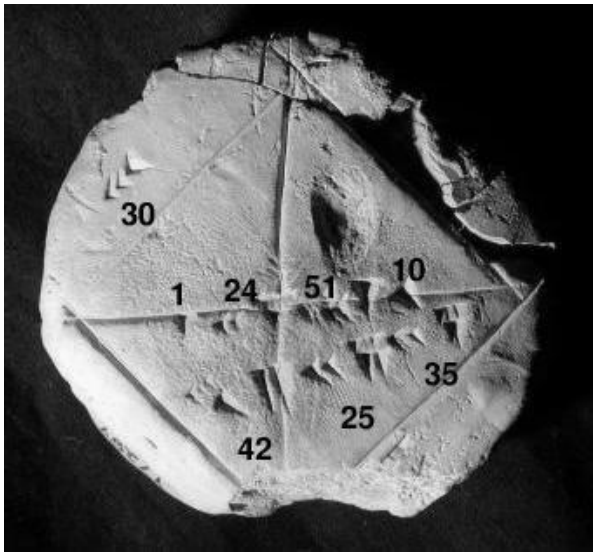
Outline

Numbers



Babylonian clay tablet (1800–1600 BC)

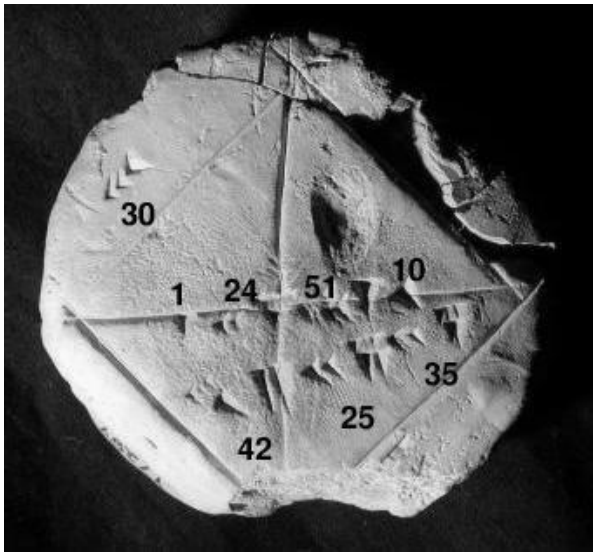
Numbers



Babylonian clay tablet (1800–1600 BC)

$$= (1, 24, 51, 10)$$
$$= 1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 30547/21600$$
$$\approx \mathbf{1,41421}96 \approx \sqrt{2}$$

Numbers



Babylonian clay tablet (1800–1600 BC)

$$\begin{aligned} \text{𒌋 𒐜 𒐜 𒐚} &= (1, 24, 51, 10) \\ &= 1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 30547/21600 \\ &\approx \mathbf{1,41421}96 \approx \sqrt{2} \end{aligned}$$

$\Rightarrow$ representation of a real number with a finite precision

Floating-Point Number

Using a finite number of bits (the precision p) and based on scientific notation, computers use floating-point (FP) numbers.

A FP number is only a string of bits.

$$11100011010010011110000111000000$$

Floating-Point Number

Using a finite number of bits (the precision p) and based on scientific notation, computers use floating-point (FP) numbers.

A FP number is only a string of bits.

$$11100011010010011110000111000000$$

We interpret it depending on the respective values of s (sign), e (exponent) and f (fraction).

1	11000110	10010011110000111000000
1	11000110	10010011110000111000000
s	e	f

Floating-Point Number

We associate a real value:

$$\underset{s}{\boxed{1}} \quad \underset{e}{\boxed{11000110}} \quad \underset{f}{\boxed{10010011110000111000000}}$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$(-1)^s \times \quad 2^{e-B} \quad \times \quad 1 \bullet f$$

$$(-1)^1 \times \quad 2^{198-127} \quad \times 1.10010011110000111000000_2$$

$$-2^{54} \times 206727 \approx -3.724 \times 10^{21}$$

Floating-Point Number

We associate a real value:

$$\underset{s}{\boxed{1}} \quad \underset{e}{\boxed{11000110}} \quad \underset{f}{\boxed{10010011110000111000000}}$$

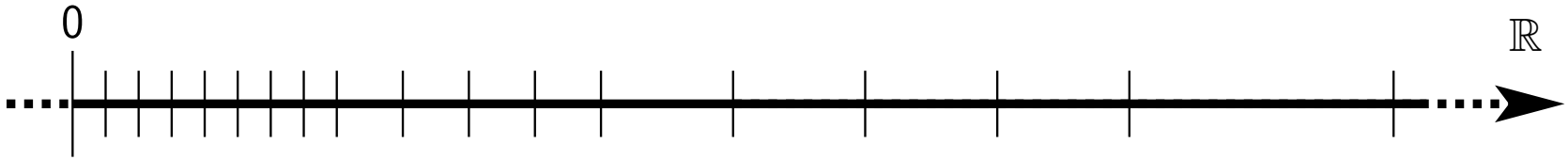
$$(-1)^{s} \times \quad 2^{e-B} \quad \times \quad 1 \bullet f$$

$$(-1)^{1} \times \quad 2^{198-127} \quad \times 1.10010011110000111000000_2$$

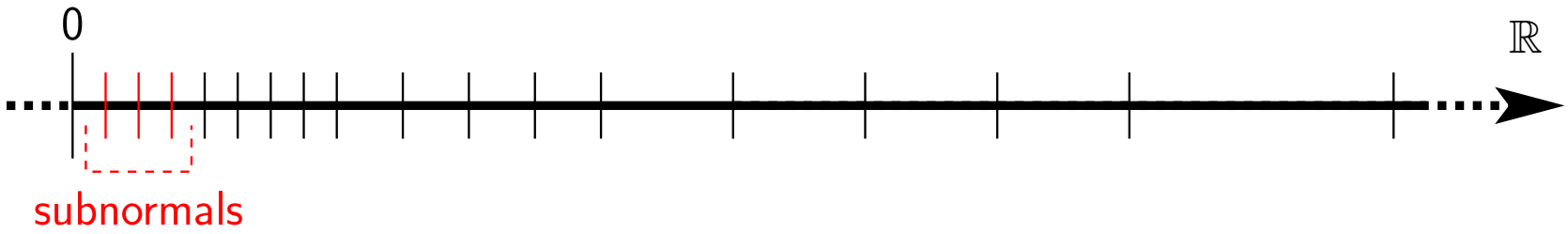
$$-2^{54} \times 206727 \approx -3.724 \times 10^{21}$$

except for the special values of e: ± 0, $\pm\infty$, NaN, subnormals.

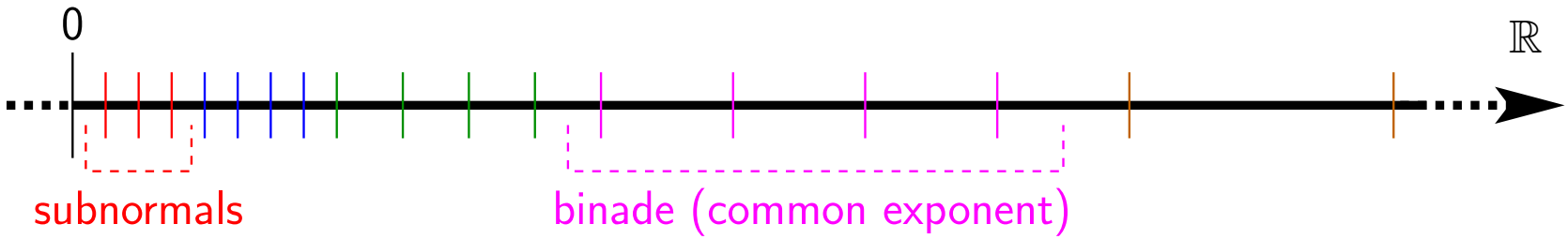
Floating-Point Number Repartition



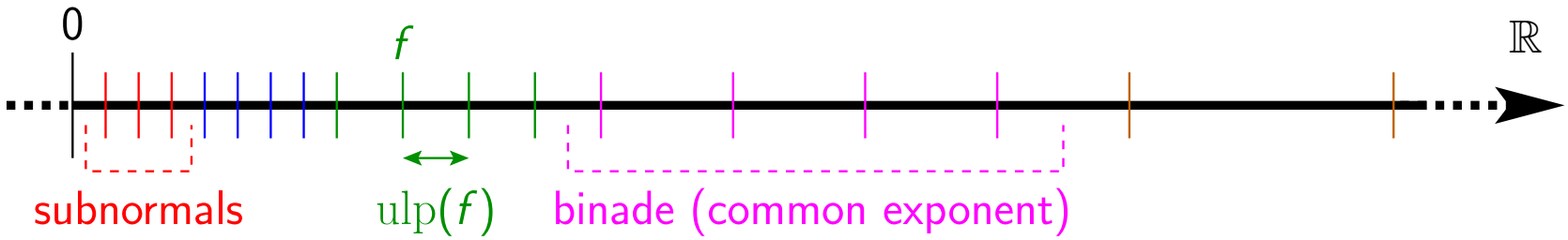
Floating-Point Number Repartition



Floating-Point Number Repartition



Floating-Point Number Repartition



Floating-Point Computation

For the $+$, $-$, $\times$, $\div$, $\sqrt{}$, the result is the same as if the infinitely precise mathematical result was computed and then rounded to the nearest floating-point number.

$\Rightarrow$ guaranteed by the IEEE-754 standard (1985 & 2008).

Floating-Point Computation

For the $+$, $-$, $\times$, $\div$, $\sqrt{}$, the result is the same as if the infinitely precise mathematical result was computed and then rounded to the nearest floating-point number.

$\Rightarrow$ guaranteed by the IEEE-754 standard (1985 & 2008).

$\Rightarrow$ portability & accuracy

Floating-Point Computation

For the $+$, $-$, $\times$, $\div$, $\sqrt{}$, the result is the same as if the infinitely precise mathematical result was computed and then rounded to the nearest floating-point number.

$\Rightarrow$ guaranteed by the IEEE-754 standard (1985 & 2008).

$\Rightarrow$ portability & accuracy

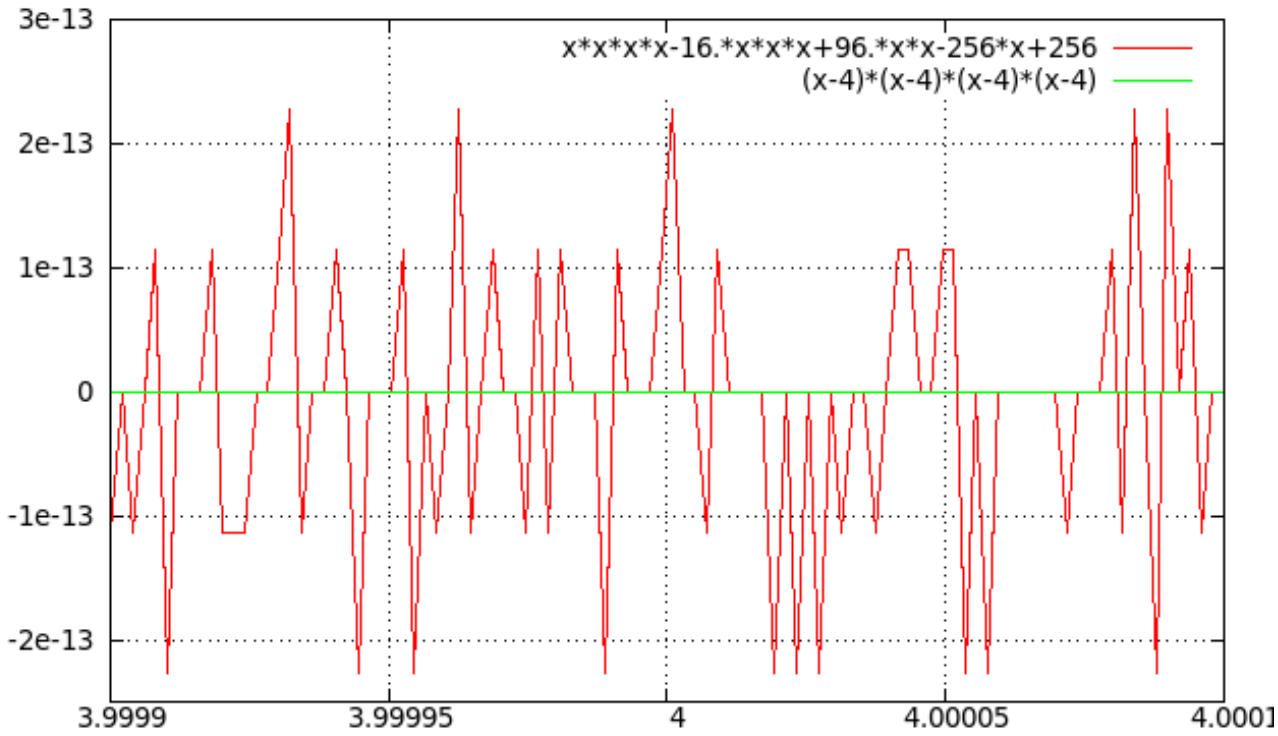
$\Rightarrow$ if $x \in \mathbb{R}$ is not too small, $|x - \circ_{\texttt{double}}(x)| \leq 2^{-53}|x|$

Floating-Point Computations

More than one FP operation may lead to incorrect results.

Floating-Point Computations

More than one FP operation may lead to incorrect results.



Floating-point evaluations of $(x-4)^4$ around 4.

Outline

Annotation language: ACSL

- ANSI/ISO C Specification Language

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs
- pre-conditions and post-conditions to functions
  (and which variables are modified).

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs
- pre-conditions and post-conditions to functions (and which variables are modified).
- variants and invariants of the loops.

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs
- pre-conditions and post-conditions to functions (and which variables are modified).
- variants and invariants of the loops.
- assertions

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs
- pre-conditions and post-conditions to functions (and which variables are modified).
- variants and invariants of the loops.
- assertions
- In annotations, all computations are exact.

Annotation language: ACSL

- ANSI/ISO C Specification Language
- behavioral specification language for C programs
- pre-conditions and post-conditions to functions (and which variables are modified).
- variants and invariants of the loops.
- assertions
- In annotations, all computations are exact.

$\Rightarrow$ **For the programmer, the specification is easy to understand.**

ACSL and floating-point numbers

A floating-point number is a triple:

- the floating-point number, really computed by the program, $x \rightarrow x_f$ floating-point part

ACSL and floating-point numbers

A floating-point number is a triple:

- the floating-point number, really computed by the program, $x \rightarrow x_f$ floating-point part
- the value that would have been obtained with exact computations, $x \rightarrow x_e$ exact part

ACSL and floating-point numbers

A floating-point number is a triple:

- the floating-point number, really computed by the program,
  $x \rightarrow x_f$ floating-point part
- the value that would have been obtained with exact computations,
  $x \rightarrow x_e$ exact part
- the value that we ideally wanted to compute
  $x \rightarrow x_m$ model part

ACSL and floating-point numbers

A floating-point number is a triple:

- the floating-point number, really computed by the program,
  $x \to x_f$ floating-point part `1+x+x*x/2`
- the value that would have been obtained with exact computations,
  $x \to x_e$ exact part $1 + x + \frac{x^2}{2}$
- the value that we ideally wanted to compute
  $x \to x_m$ model part $\exp(x)$

ACSL and floating-point numbers

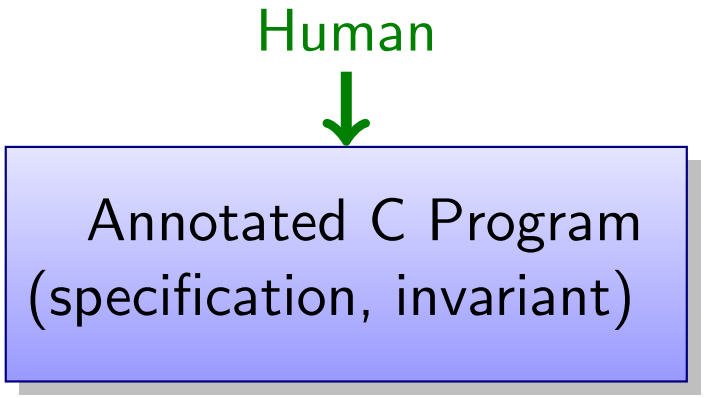
A floating-point number is a triple:

- the floating-point number, really computed by the program,
  $x \rightarrow x_f$ floating-point part `1+x+x*x/2`
- the value that would have been obtained with exact computations,
  $x \rightarrow x_e$ exact part $1 + x + \frac{x^2}{2}$
- the value that we ideally wanted to compute
  $x \rightarrow x_m$ model part $\exp(x)$

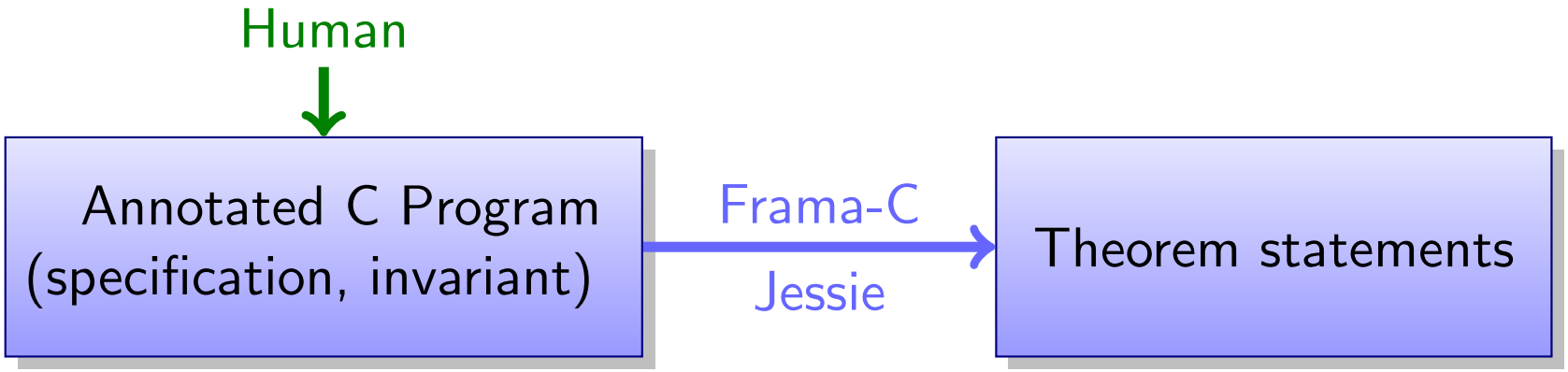
$\Rightarrow$ easy to split into method error and rounding error

Methodology for the verification of C programs

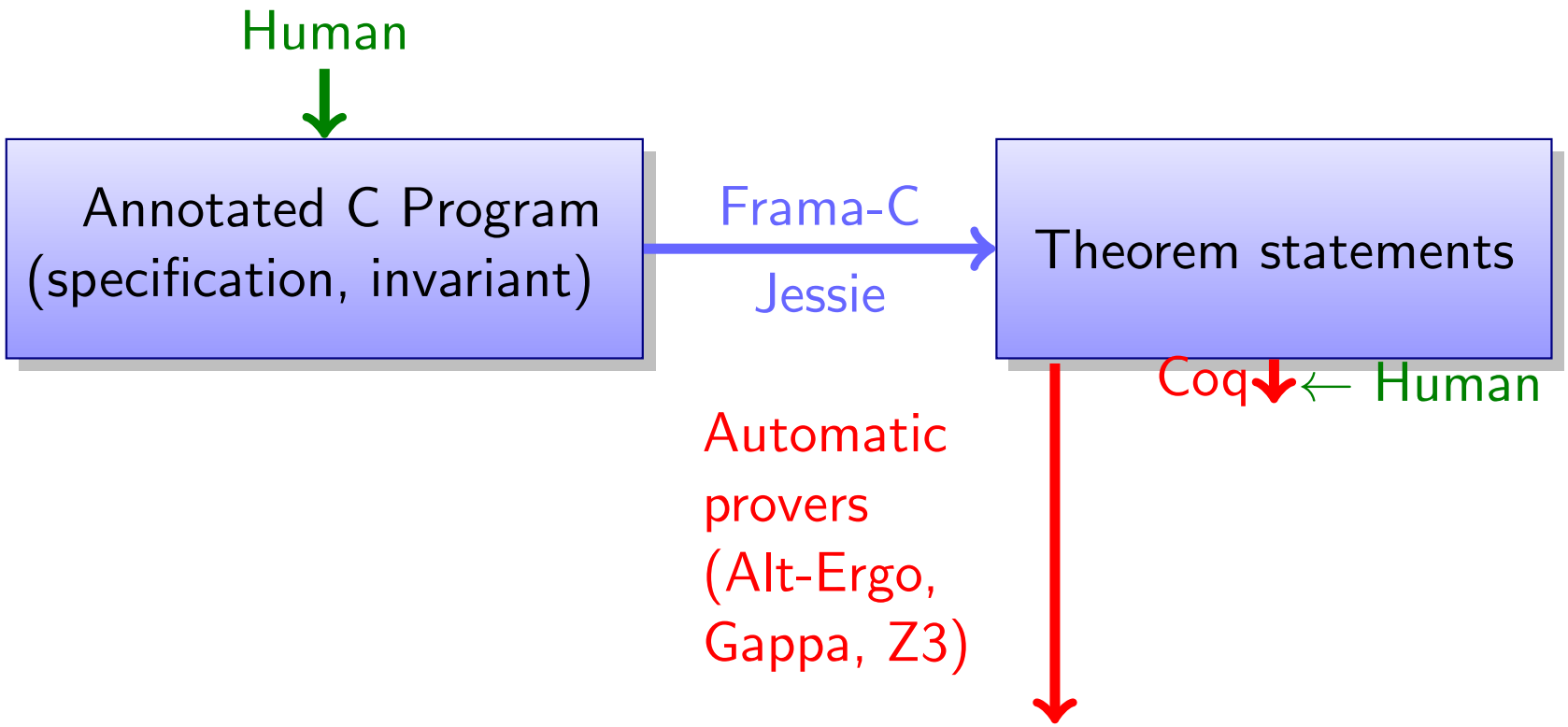
Methodology for the verification of C programs



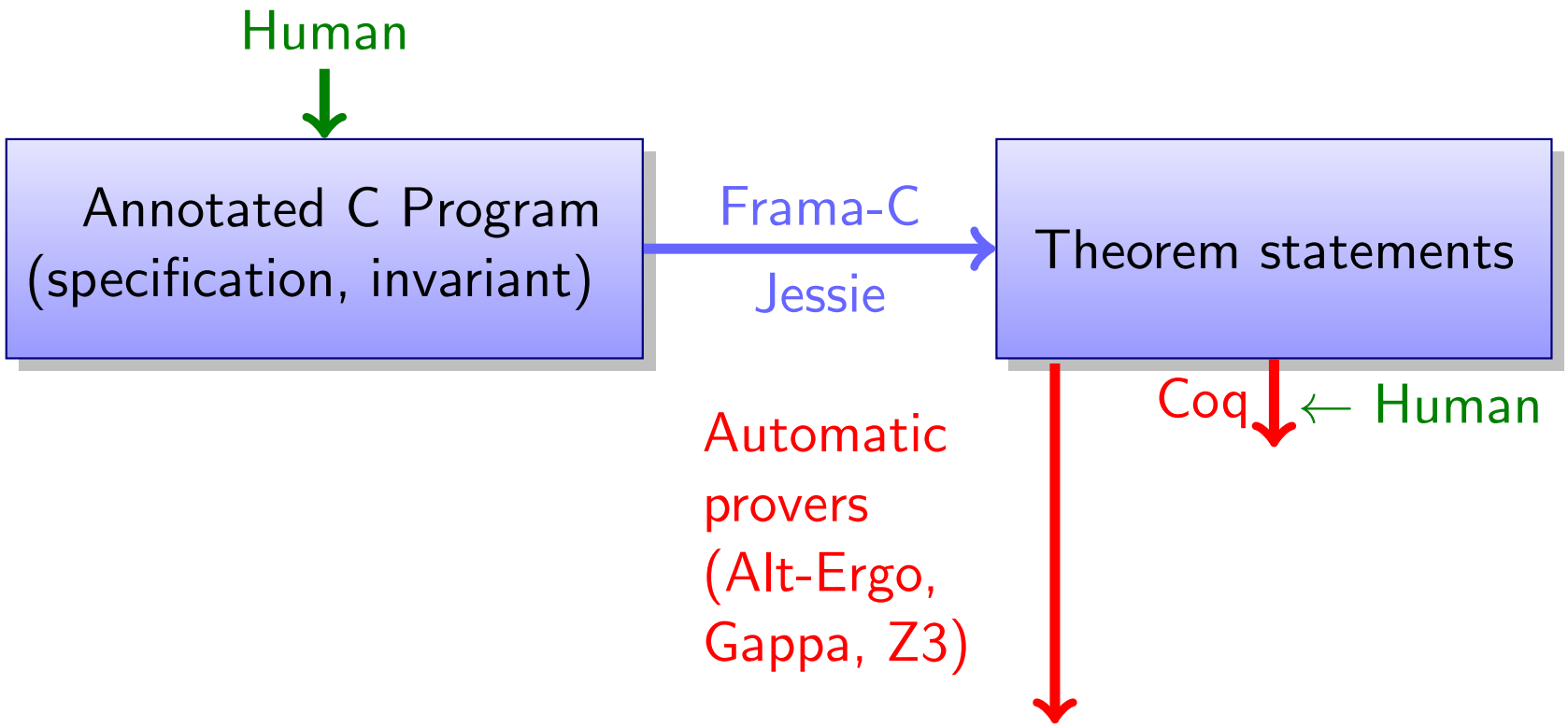
Methodology for the verification of C programs



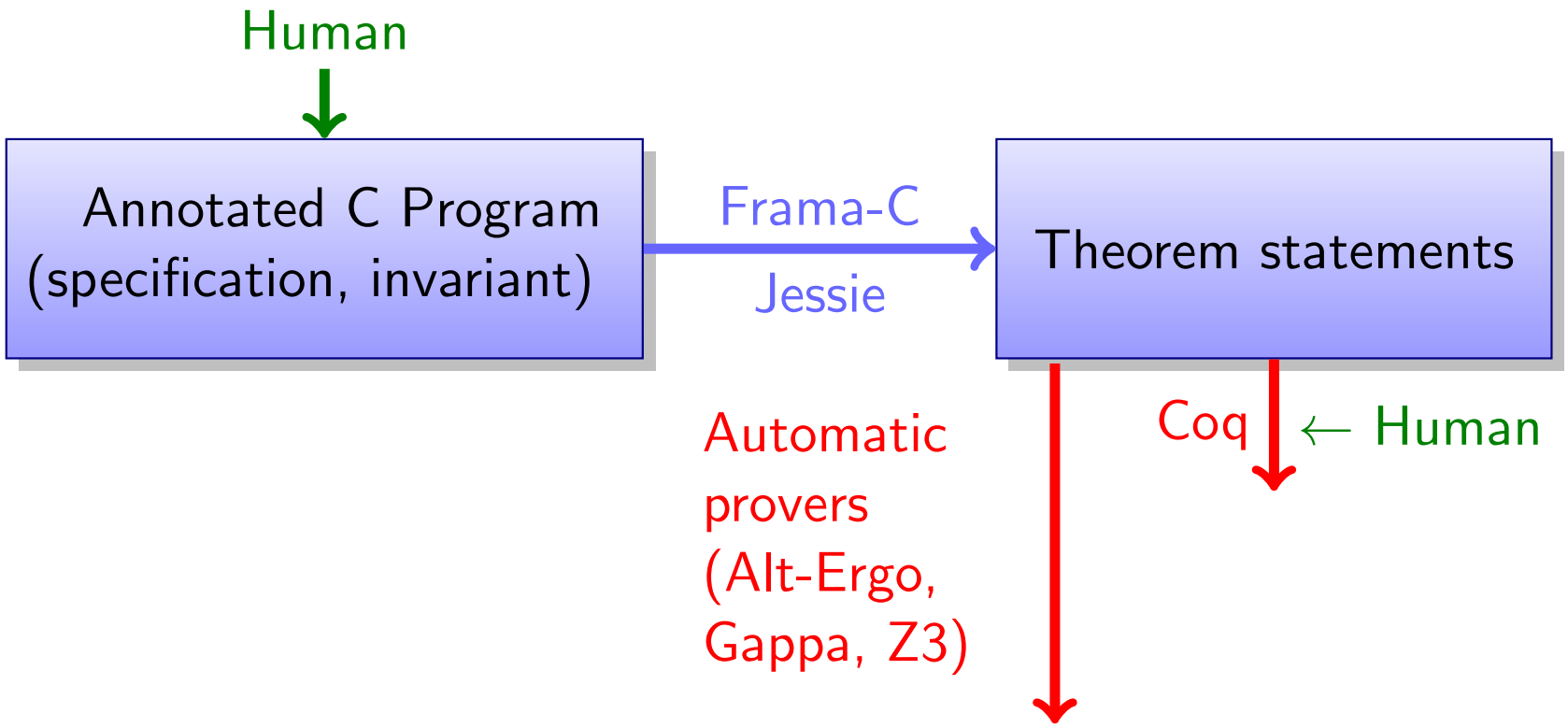
Methodology for the verification of C programs



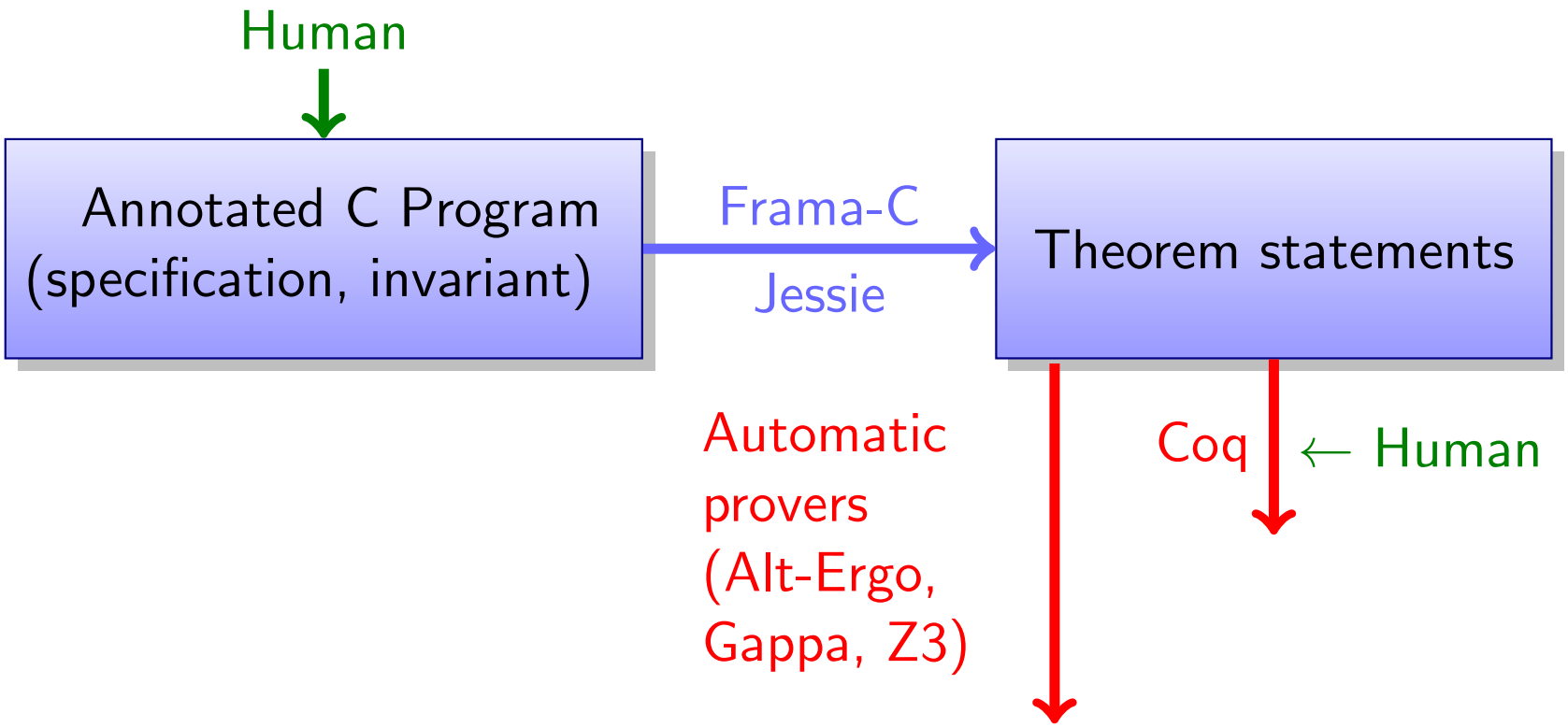
Methodology for the verification of C programs



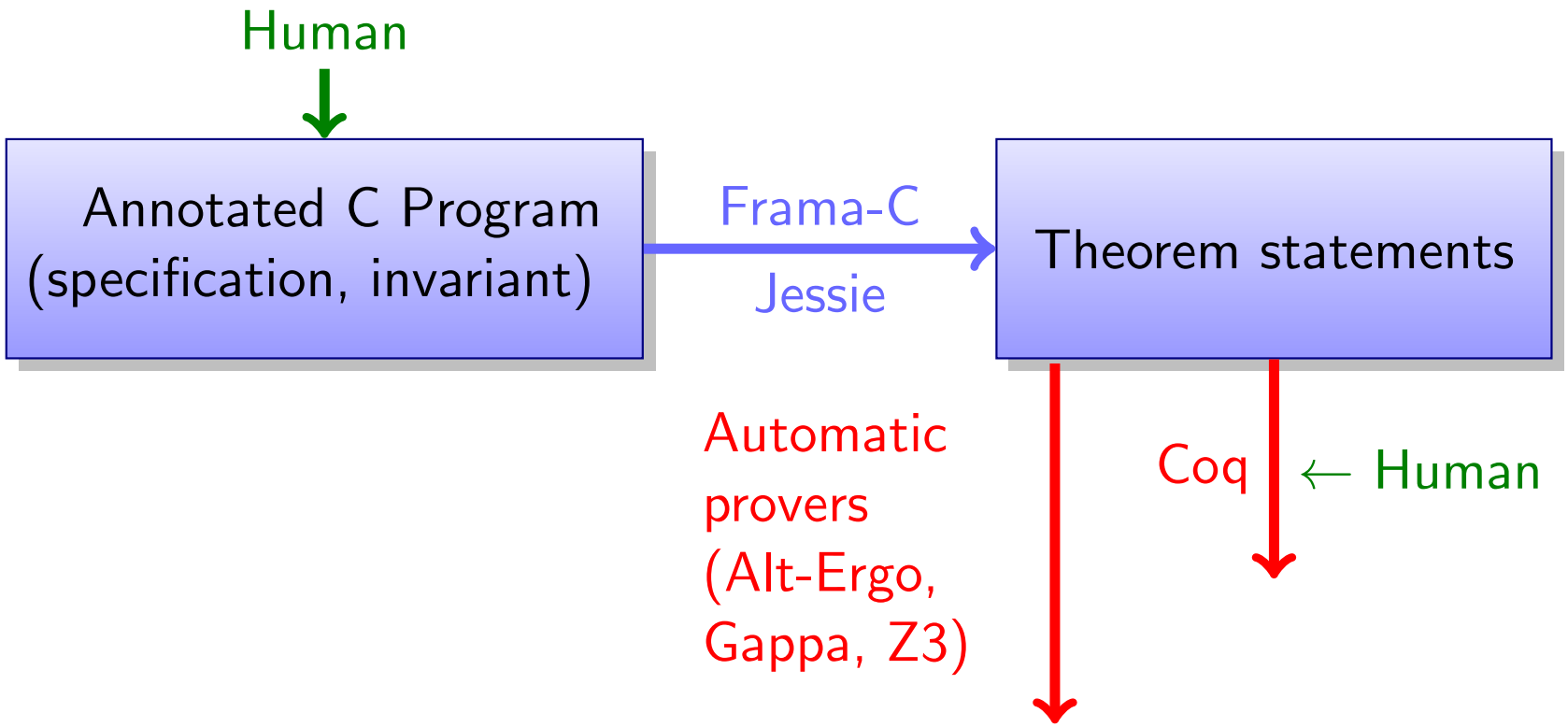
Methodology for the verification of C programs



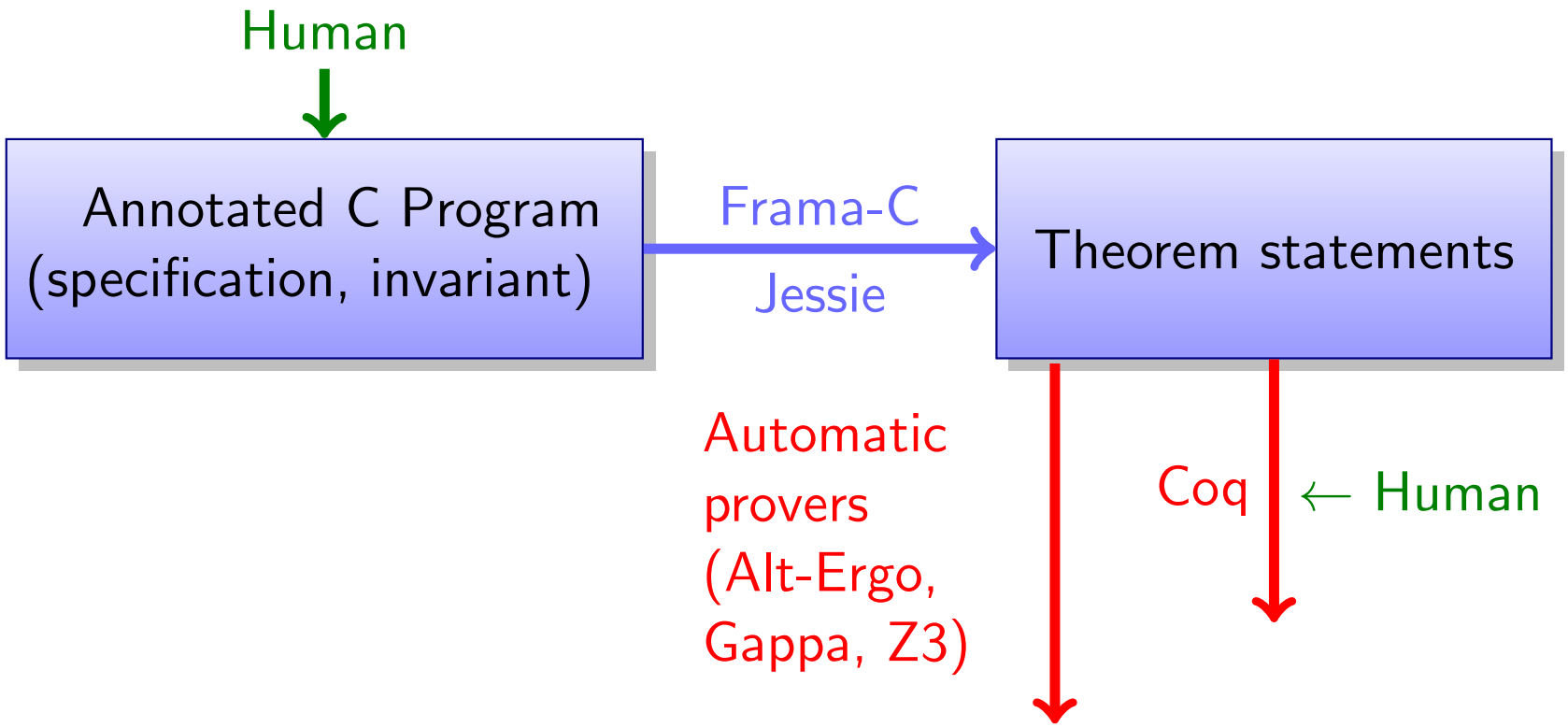
Methodology for the verification of C programs



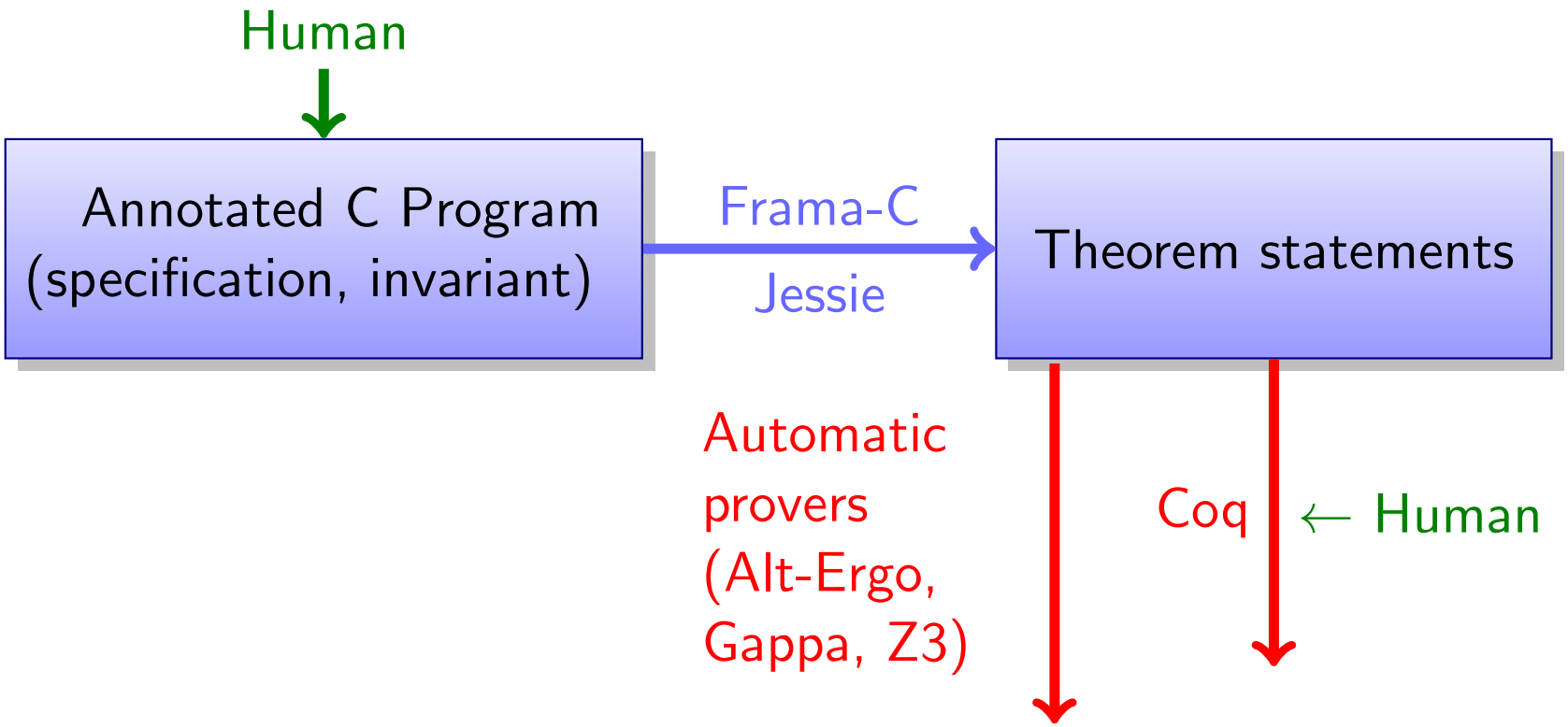
Methodology for the verification of C programs



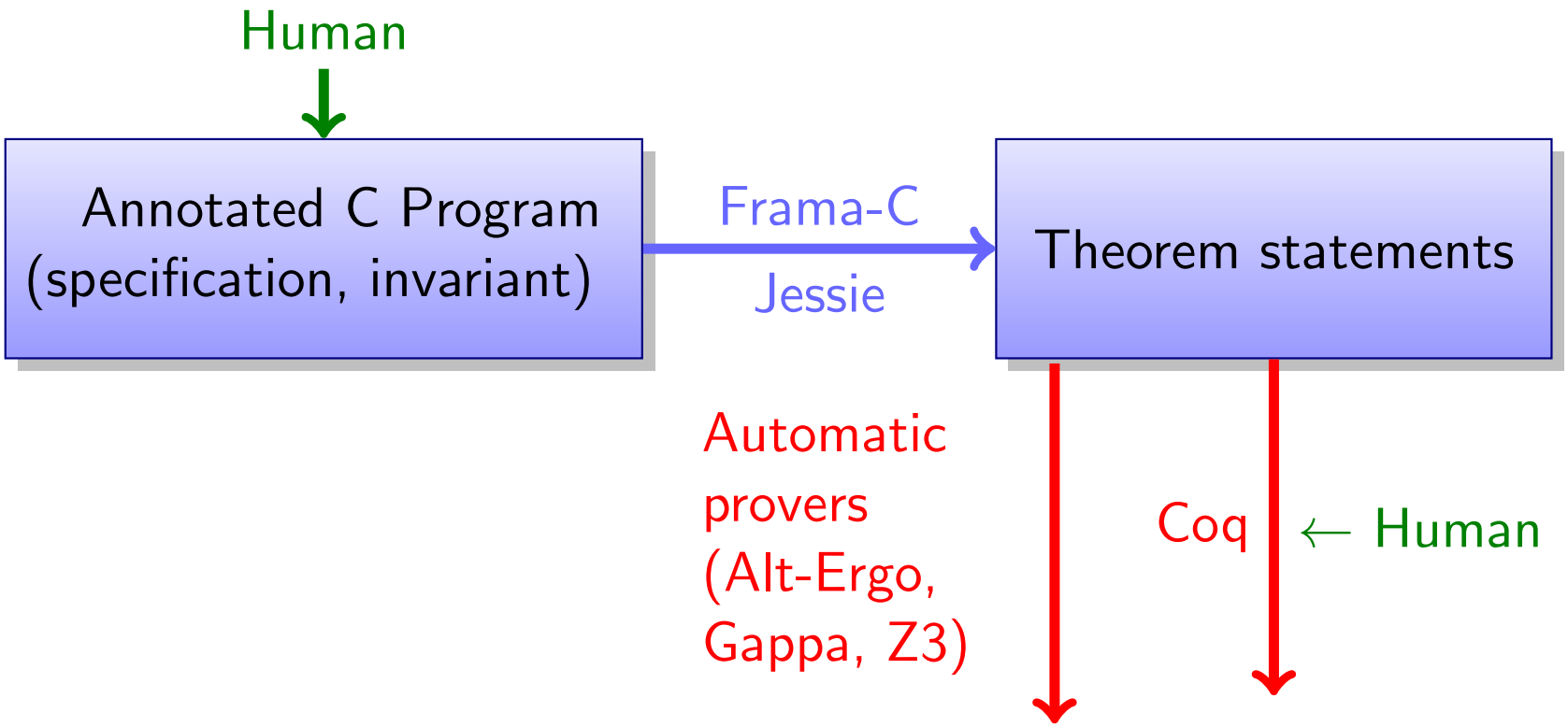
Methodology for the verification of C programs



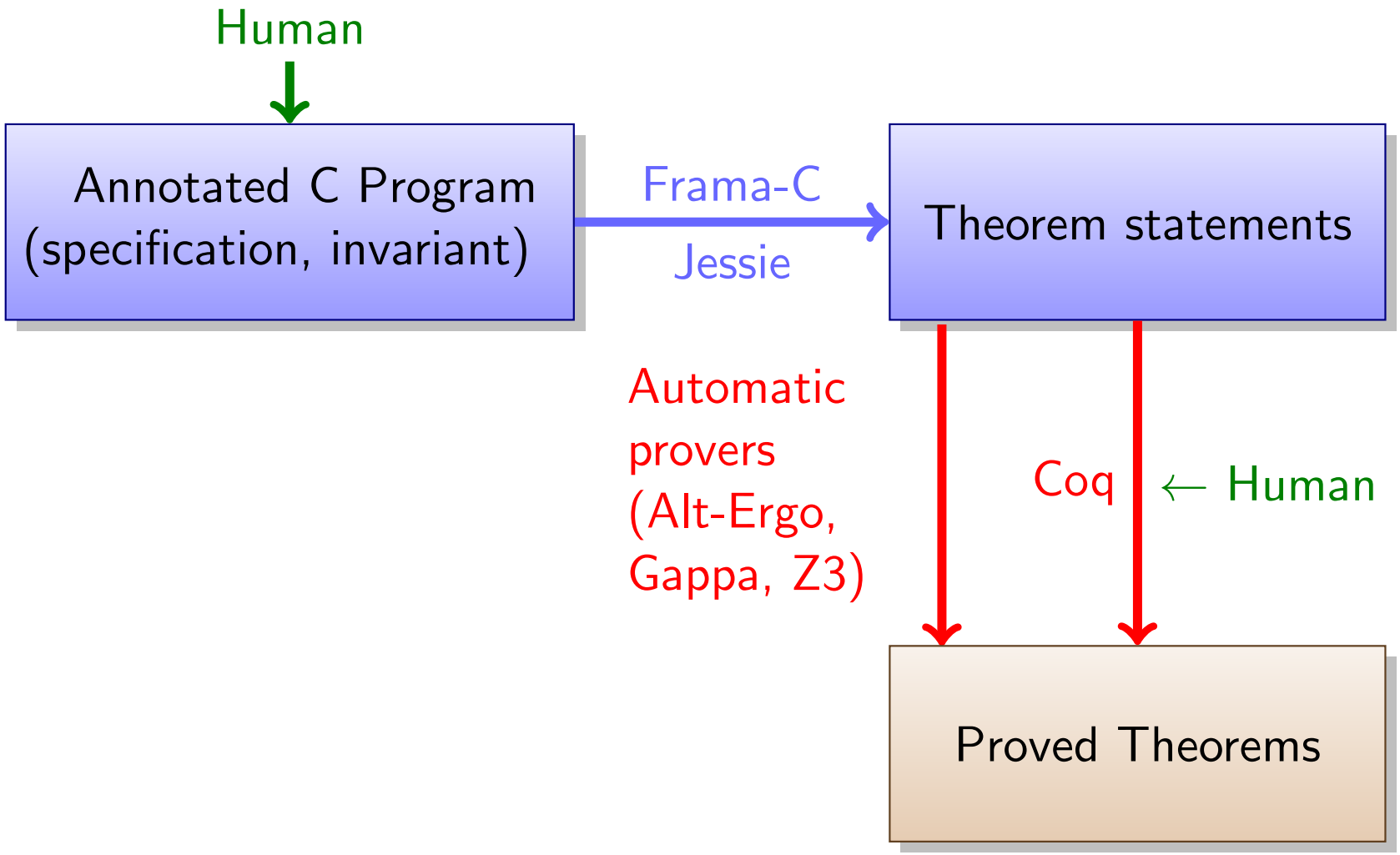
Methodology for the verification of C programs



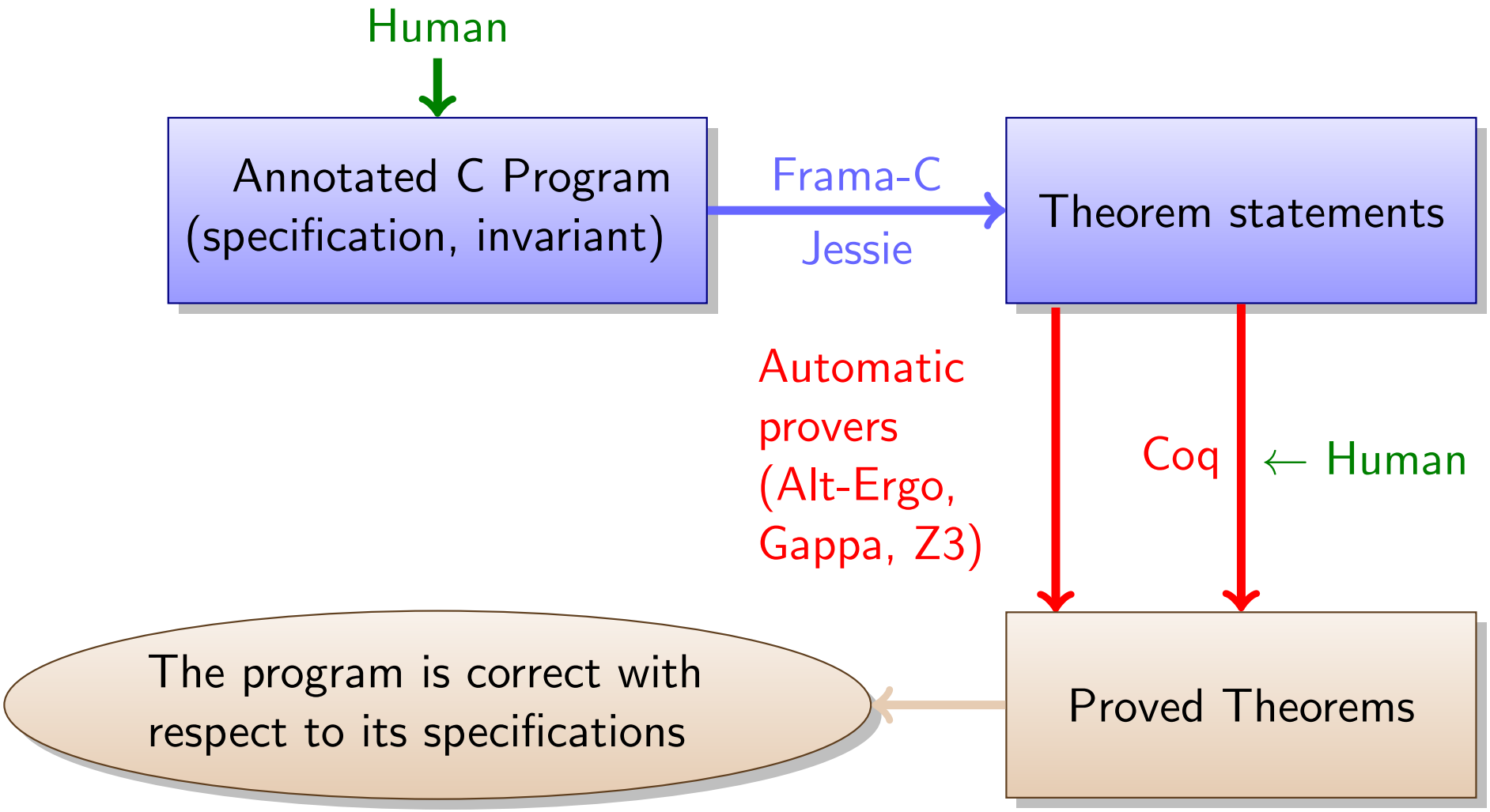
Methodology for the verification of C programs



Methodology for the verification of C programs



Methodology for the verification of C programs



Outline

The wave equation

Looking for $u : \mathbb{R}^2 \to \mathbb{R}$ regular enough such that:

$$\frac{\partial^2 u(x,t)}{\partial t^2} - c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = s(x,t)$$

with given values for the initial position $u_0(x)$ and the initial velocity $u_1(x)$.

$\Rightarrow$ rope oscillation, sound, radar, oil prospection...

Scheme?

We want $u_j^k \approx u(j\Delta x, k\Delta t)$.

$$\frac{u_j^k - 2u_j^{k-1} + u_j^{k-2}}{\Delta t^2} - c^2 \frac{u_{j+1}^{k-1} - 2u_j^{k-1} + u_{j-1}^{k-1}}{\Delta x^2} = s_j^{k-1}$$

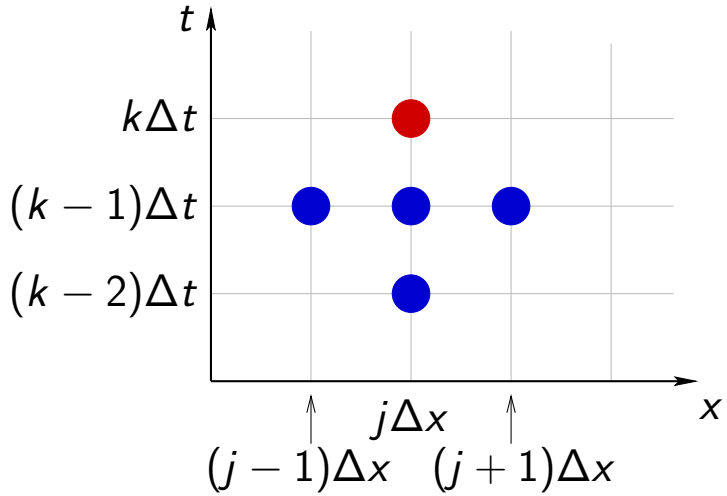
And other horrible formulas to initialize u_j^0 and u_j^1.

Scheme?

We want $u_j^k \approx u(j\Delta x, k\Delta t)$.

$$\frac{u_j^k - 2u_j^{k-1} + u_j^{k-2}}{\Delta t^2} - c^2 \frac{u_{j+1}^{k-1} - 2u_j^{k-1} + u_{j-1}^{k-1}}{\Delta x^2} = s_j^{k-1}$$

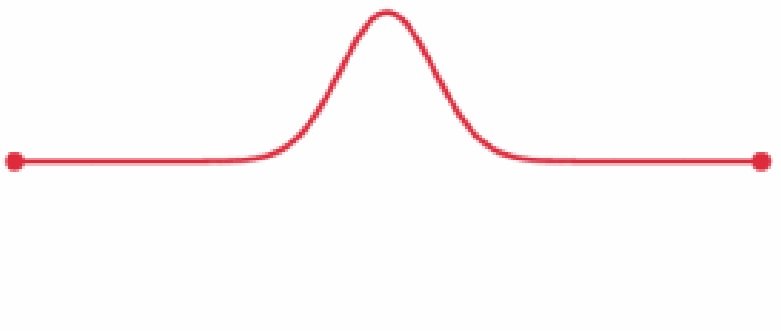
And other horrible formulas to initialize u_j^0 and u_j^1.



Three-point scheme: u_j^k depends on u_{j-1}^{k-1}, u_j^{k-1}, u_{j+1}^{k-1} and u_j^{k-2}.

Program

```
// initialization of p[i][0] and p[i][1]
for (k=1; k<nk; k++) {
  p[0][k+1] = 0.;
  for (i=1; i<ni; i++) {
    dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
    p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
   }
  p[ni][k+1] = 0.;
}
```



Program

```
// initialization of p[i][0] and p[i][1]
for (k=1; k<nk; k++) {
  p[0][k+1] = 0.;
  for (i=1; i<ni; i++) {
    dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
    p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
  }
  p[ni][k+1] = 0.;
}
```

Two different errors:

- round-off errors
  due to floating-point roundings
- method errors
  the scheme only approximates the exact solution

Outline

Rounding error

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

If we use a naive technique to bound the rounding errors, we get

Rounding error

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

If we use a naive technique to bound the rounding errors, we get

$$|p_i^k - exact(p_i^k)| \leq O\left(2^k 2^{-53}\right)$$

Rounding error

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

If we use a naive technique to bound the rounding errors, we get

$$|p_i^k - exact(p_i^k)| \leq O\left(2^k 2^{-53}\right)$$

This is too much because the errors do compensate.

Definition of ε_i^k

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

Let ε_i^{k+1} be the rounding error made during these two lines of computations.

We assume a, p_{i-1}^k, p_i^k, p_{i+1}^k and p_i^{k-1} are exact and we look into the rounding error of these two lines. It is called ε_i^{k+1}.

Definition of ε_i^k

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

Let ε_i^{k+1} be the rounding error made during these two lines of computations.

We assume a, p_{i-1}^k, p_i^k, p_{i+1}^k and p_i^{k-1} are exact and we look into the rounding error of these two lines. It is called ε_i^{k+1}.

We know (from initializations) that the model values of the $|p_n^m|$ are bounded by 1. We assume that the floating-point values of the $|p_n^m|$ are bounded by 2.

Definition of ε_i^k

Remainder:

```
dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
```

Let ε_i^{k+1} be the rounding error made during these two lines of computations.

We assume a, p_{i-1}^k, p_i^k, p_{i+1}^k and p_i^{k-1} are exact and we look into the rounding error of these two lines. It is called ε_i^{k+1}.

We know (from initializations) that the model values of the $|p_n^m|$ are bounded by 1. We assume that the floating-point values of the $|p_n^m|$ are bounded by 2.

$$|\varepsilon_n^m| \leq 78 \times 2^{-52}$$

Rounding error

$$p_i^k - exact(p_i^k) = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_{i+j}^{k-l}$$

1. We have an analytical expression of the rounding error with known constants α_i^k.

Rounding error

$$p_i^k - exact(p_i^k) = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_{i+j}^{k-l}$$

1. We have an analytical expression of the rounding error with known constants α_i^k.
2. It is not that complicated!
   (we cannot get rid of the pyramidal double summation)

Rounding error

$$p_i^k - exact(p_i^k) = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_{i+j}^{k-l}$$

1. We have an analytical expression of the rounding error with known constants α_i^k.
2. It is not that complicated!
   (we cannot get rid of the pyramidal double summation)
3. The rounding error is bounded by $\bigcirc(k^2\ 2^{-53})$:

$$\left|\mathbf{p_i^k} - \mathbf{exact}\left(\mathbf{p_i^k}\right)\right| \leq \mathbf{78} \times \mathbf{2^{-53}} \times (\mathbf{k}+\mathbf{1}) \times (\mathbf{k}+\mathbf{2})$$

Outline

Method error

We measure that u and u_j^k are close when $(\Delta x, \Delta t) \to 0$.

We define $e_j^k \stackrel{\text{def}}{=} \bar{u}_j^k - u_j^k$: convergence error
where $\bar{u}_j^k$ is the value of u at the (j, k) point of the grid.

Method error

We measure that u and u_j^k are close when $(\Delta x, \Delta t) \to 0$.

We define $e_j^k \stackrel{\text{def}}{=} \bar{u}_j^k - u_j^k$: convergence error
where $\bar{u}_j^k$ is the value of u at the (j,k) point of the grid.

We want to bound $\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x}$: the average of the convergence error on all points of the grid at a given time $k_{\Delta t}(t) = \left\lfloor \frac{t}{\Delta t} \right\rfloor \Delta t$.

Method error

We measure that u and u_j^k are close when $(\Delta x, \Delta t) \to 0$.

We define $e_j^k \stackrel{\text{def}}{=} \bar{u}_j^k - u_j^k$: convergence error
where $\bar{u}_j^k$ is the value of u at the (j, k) point of the grid.

We want to bound $\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x}$: the average of the convergence error on all points of the grid at a given time $k_{\Delta t}(t) = \left\lfloor \frac{t}{\Delta t} \right\rfloor \Delta t$.

We want to prove:

$$\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x} = O_{[0,t_{\max}]}(\Delta x^2 + \Delta t^2)$$

Big O = big pain

Usually, the big O uses one variable and $f(\mathbf{x}) = O_{\|\mathbf{x}\|\to 0}(g(\mathbf{x}))$ means

$$\exists \alpha, C > 0, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad \|\mathbf{x}\| \leq \alpha \Rightarrow |f(\mathbf{x})| \leq C \cdot |g(\mathbf{x})|.$$

Big O = big pain

Usually, the big O uses one variable and $f(\mathbf{x}) = O_{\|\mathbf{x}\|\to 0}(g(\mathbf{x}))$ means

$$\exists \alpha, C > 0, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad \|\mathbf{x}\| \leq \alpha \Rightarrow |f(\mathbf{x})| \leq C \cdot |g(\mathbf{x})|.$$

Here 2 variables: $\boldsymbol{\Delta}\mathbf{x}$ (grid sizes, tends to 0), and $\mathbf{x}$ (time and space).
(Think about Taylor expansions)

Big O = big pain

Usually, the big O uses one variable and $f(\mathbf{x}) = O_{\|\mathbf{x}\|\to 0}(g(\mathbf{x}))$ means

$$\exists \alpha, C > 0, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad \|\mathbf{x}\| \leq \alpha \Rightarrow |f(\mathbf{x})| \leq C \cdot |g(\mathbf{x})|.$$

Here 2 variables: $\mathbf{\Delta x}$ (grid sizes, tends to 0), and $\mathbf{x}$ (time and space).
(Think about Taylor expansions)

$$\forall \mathbf{x}, \exists \alpha, C > 0, \quad \forall \mathbf{\Delta x} \in \mathbb{R}^2, \quad \|\mathbf{\Delta x}\| \leq \alpha \Rightarrow |f(\mathbf{x}, \mathbf{\Delta x})| \leq C \cdot |g(\mathbf{\Delta x})|$$

does not work.

Uniform big O

We used a uniform big O:

$$\exists \alpha, C > 0, \quad \forall \mathbf{x}, \mathbf{\Delta x}, \quad \|\mathbf{\Delta x}\| \leq \alpha \Rightarrow |f(\mathbf{x}, \mathbf{\Delta x})| \leq C \cdot |g(\mathbf{\Delta x})|.$$

where variables $\mathbf{x}$ and $\mathbf{\Delta x}$ are restricted to subsets of $\mathbb{R}^2$.
(for example such that $\Delta t > 0$)
$\Rightarrow$ Taylor expansions

Proof idea 1/3: consistency

The truncation error is defined as how much the exact solution solves the numerical scheme:

$$\varepsilon_j^{k-1} = \frac{\bar{u}_j^k - 2\bar{u}_j^{k-1} + \bar{u}_j^{k-2}}{\Delta t^2} - c^2 \frac{\bar{u}_{j+1}^{k-1} - 2\bar{u}_j^{k-1} + \bar{u}_{j-1}^{k-1}}{\Delta x^2} - s_j^{k-1}$$

Proof idea 1/3: consistency

The truncation error is defined as how much the exact solution solves the numerical scheme:

$$\varepsilon_j^{k-1} = \frac{\bar{u}_j^k - 2\bar{u}_j^{k-1} + \bar{u}_j^{k-2}}{\Delta t^2} - c^2 \frac{\bar{u}_{j+1}^{k-1} - 2\bar{u}_j^{k-1} + \bar{u}_{j-1}^{k-1}}{\Delta x^2} - s_j^{k-1}$$

The consistency is the boundedness of the truncation error:

$$\left\| \varepsilon_h^{k_{\Delta t}(t)} \right\|_{\Delta x} = O_{[0,t_{\max}]}(\Delta x^2 + \Delta t^2)$$

By Taylor series and many computations.

Proof idea 2/3: stability

We define a discrete energy by

$$E_h(c)(u_h)^{k+\frac{1}{2}} \stackrel{\text{def}}{=} \frac{1}{2}\left\|\frac{u_h^{k+1}-u_h^k}{\Delta t}\right\|_{\Delta x}^2 + \frac{1}{2}\left\langle u_h^k, u_h^{k+1}\right\rangle_{A_h(c)}$$

kinetic energy potential energy

$$\left\langle v_h, w_h\right\rangle_{A_h(c)} \stackrel{\text{def}}{=} \left\langle A_h(c)\, v_h, w_h\right\rangle_{\Delta x} \text{ and } \left(A_h(c)\, v_h\right)_j \stackrel{\text{def}}{=} -c^2\frac{v_{j+1}-2v_j+v_{j-1}}{\Delta x^2}.$$

Proof idea 2/3: stability

We define a discrete energy by

$$E_h(c)(u_h)^{k+\frac{1}{2}} \stackrel{\text{def}}{=} \frac{1}{2}\left\|\frac{u_h^{k+1}-u_h^k}{\Delta t}\right\|_{\Delta x}^2 + \frac{1}{2}\left\langle u_h^k, u_h^{k+1}\right\rangle_{A_h(c)}$$

kinetic energy potential energy

$\left\langle v_h, w_h\right\rangle_{A_h(c)} \stackrel{\text{def}}{=} \left\langle A_h(c)\, v_h, w_h\right\rangle_{\Delta x}$ and $\left(A_h(c)\, v_h\right)_j \stackrel{\text{def}}{=} -c^2 \frac{v_{j+1}-2v_j+v_{j-1}}{\Delta x^2}$.

Note that this energy is constant if $f = 0$.
We prove an overestimation and an underestimation of this energy.
$\Rightarrow$ u_h does not diverge.

Proof idea 3/3: convergence

The convergence error is solution of the same discrete scheme with inputs

$$u_{0,j} = 0, \qquad u_{1,j} = \frac{e_j^1}{\Delta t}, \qquad \text{and} \qquad s_j^k = \varepsilon_j^{k+1}.$$

+ proofs about the initializations.

Proof idea 3/3: convergence

The convergence error is solution of the same discrete scheme with inputs

$$u_{0,j} = 0, \qquad u_{1,j} = \frac{e_j^1}{\Delta t}, \qquad \text{and} \qquad s_j^k = \varepsilon_j^{k+1}.$$

+ proofs about the initializations.

All these proofs require the existence of ζ and ξ in $]0,1[$ with $\zeta \leq 1 - \xi$ and we require that $\zeta \leq \frac{c\Delta t}{\Delta x} \leq 1 - \xi$ (CFL conditions).

Convergence

We proved that:

$$\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x} = O\left(\begin{array}{l} t \in [0, t_{\max}] \\ (\Delta x, \Delta t) \to 0 \\ 0 < \Delta x \wedge 0 < \Delta t \wedge \\ \zeta \leq c\frac{\Delta t}{\Delta x} \leq 1 - \xi \end{array} \right. \quad (\Delta x^2 + \Delta t^2).$$

Extraction of the big O constants

The preceding result is a uniform big O defined by:

$$\exists \alpha, C > 0, \quad \forall \mathbf{x}, \mathbf{\Delta x}, \quad \|\mathbf{\Delta x}\| \leq \alpha \Rightarrow |f(\mathbf{x}, \mathbf{\Delta x})| \leq C \cdot |g(\mathbf{\Delta x})|.$$

Extraction of the big O constants

The preceding result is a uniform big O defined by:

$$\exists \alpha, C > 0, \quad \forall \mathbf{x}, \mathbf{\Delta x}, \quad \|\mathbf{\Delta x}\| \leq \alpha \Rightarrow |f(\mathbf{x}, \mathbf{\Delta x})| \leq C \cdot |g(\mathbf{\Delta x})|.$$

Let (α_3, C_3) be the constants for the order-3 Taylor development of the exact solution and (α_4, C_4) for order-4. The initial support is $[\chi_1; \chi_2]$.

Extraction of the big O constants

The preceding result is a uniform big O defined by:

$$\exists \alpha, C > 0, \quad \forall \mathbf{x}, \mathbf{\Delta x}, \quad \|\mathbf{\Delta x}\| \leq \alpha \Rightarrow |f(\mathbf{x}, \mathbf{\Delta x})| \leq C \cdot |g(\mathbf{\Delta x})|.$$

Let (α_3, C_3) be the constants for the order-3 Taylor development of the exact solution and (α_4, C_4) for order-4. The initial support is $[\chi_1; \chi_2]$.

$$\begin{aligned}
\alpha &= \min(\alpha_3, \alpha_4, 1, t_{\max}) \\
s_1 &= \max(1, 2 \cdot C_4 \cdot (c^2 + 1), C_3 \cdot (1 + c^2/2) + 1) \\
s_2 &= s_1^2 \left(\lfloor \chi_2 \rfloor - \lfloor \chi_1 \rfloor + 2 \cdot c \cdot t_{\max} \cdot \left(1 + \frac{1}{\zeta} \right) + 3 \right) \\
s_3 &= \frac{1}{\sqrt{2}} \left(C_3 \cdot (1 + c^2/2) + 1 \right) \cdot (\chi_2 - \chi_1 + 1 + (2 \cdot c + 4)) \\
&\quad + \frac{\sqrt{2}}{2\sqrt{2\xi - \xi^2}} (2 \cdot t_{\max} \cdot s_2 + 2 s_2) \\
C &= \frac{\sqrt{2}}{\sqrt{2\xi - \xi^2}} \cdot 2 \cdot t_{\max} \cdot s_3
\end{aligned}$$

Outline

Program verification

- 154 lines of annotations for 32 lines of C
- 150 verification conditions:
  - 44 about the behavior
  - 106 about the safety (runtime errors)

Program verification

- 154 lines of annotations for 32 lines of C
- 150 verification conditions:
  - 44 about the behavior
  - 106 about the safety (runtime errors)

Prover	Behavior VC	Safety VC	Total
Alt-Ergo	18	80	98
CVC3	18	89	107
Gappa	2	20	22
Z3	21	63	84
Automatically proved	**23**	**94**	**117**
Coq	**21**	**12**	**33**
Total	44	106	150

Program verification

- About 90 % of the safety goals (matrix access, Overflow, and so on) are proved automatically.
- 33 theorems are interactively proved using Coq for a total of about 15,000 lines of Coq and 30 minutes of compilation.

Type of proofs	Nb spec lines	Nb lines	Compilation time
Convergence	991	5 275	42 s
Round-off + runtime errors	7 737	13 175	32 min

Outline

Motivations

http://www.ima.umn.edu/~arnold/disasters/sleipner.html

The sinking of the Sleipner A offshore platform

Excerpted from a report of SINTEF, Civil and Environmental Engineering:

The Sleipner A platform produces oil and gas in the North Sea and is supported on the seabed at a water depth of 82 m. It is a Condeep type platform with a concrete gravity base structure consisting of 24 cells and with a total base area of 16 000 m^2. Four cells are elongated to shafts supporting the platform deck. The first concrete base structure for Sleipner A sprang a leak and sank under a controlled ballasting operation during preparation for deck mating in Gandsfjorden outside Stavanger, Norway on 23 August 1991.

Immediately after the accident, the owner of the platform, Statoil, a Norwegian oil company appointed an investigation group, and SINTEF was contracted to be the technical advisor for this group.

The investigation into the accident is described in 16 reports...

The conclusion of the investigation was that the loss was caused by a failure in a cell wall, resulting in a serious crack and a leakage that the pumps were not able to cope with. The wall failed as a result of a combination of a serious error in the finite element analysis and insufficient anchorage of the reinforcement in a critical zone.

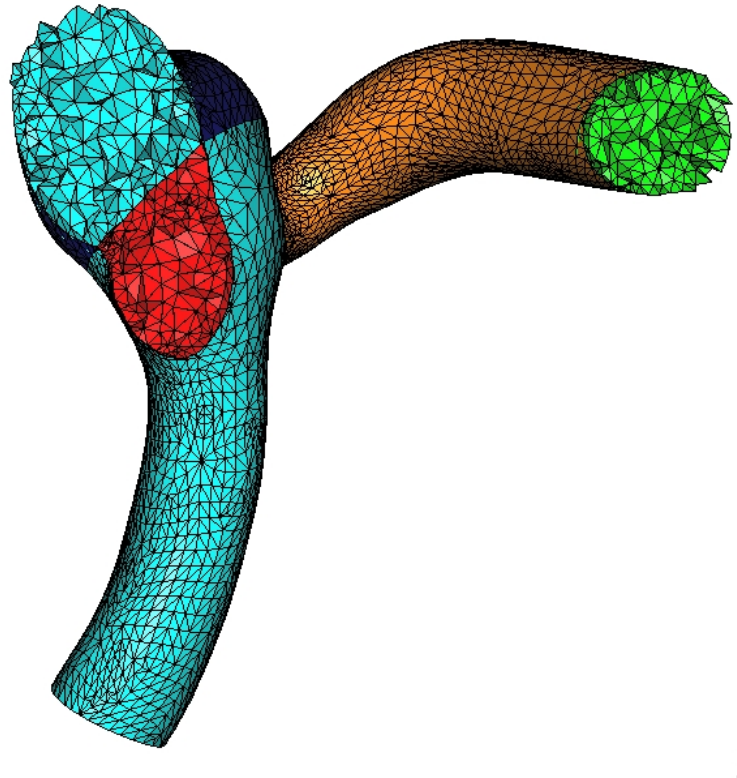
A better idea of what was involved can be obtained from this photo and sketch of the platform. The top deck weighs 57,000 tons, and provides accommodation for about 200 people and support for drilling equipment weighing about 40,000 tons. When the first model sank in August 1991, the crash

Motivations

Real life applications need solving PDE (Partial Differential Equation) on complex 3D geometries.

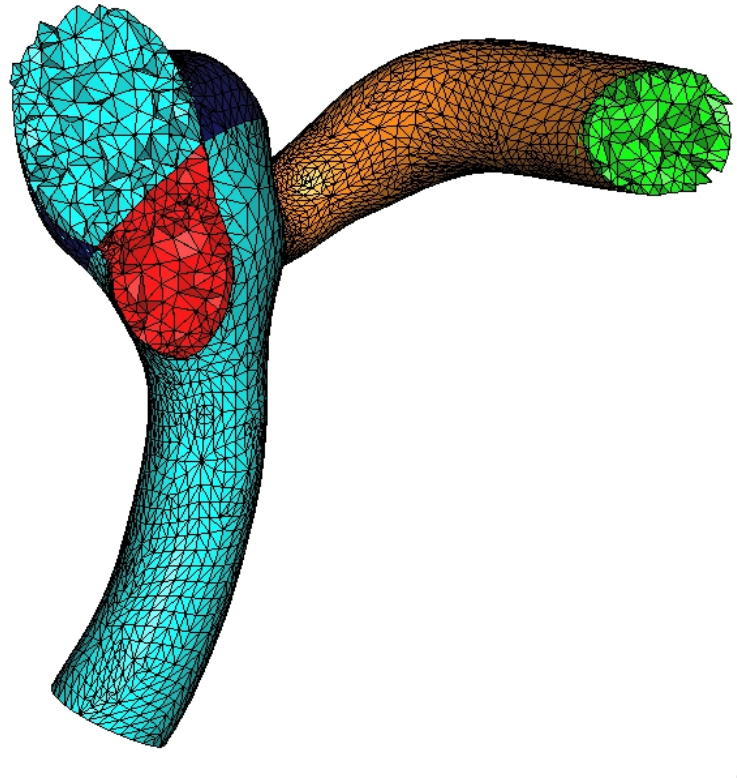
Motivations

Real life applications need solving PDE (Partial Differential Equation) on complex 3D geometries.



Motivations

Real life applications need solving PDE (Partial Differential Equation) on complex 3D geometries.



Instead of regular 2D/3D grids, we consider meshes made of triangles/tetrahedra.

Motivations

The Finite Element Method (FEM) is the most used method to solve PDEs over meshes.

> *FEM encompasses methods for connecting many simple element equations over many small subdomains, named finite elements, to approximate a more complex equation over a larger domain.*
>
> (`https://en.wikipedia.org/wiki/Finite_element_method`)

Motivations

The Finite Element Method (FEM) is the most used method to solve PDEs over meshes.

> *FEM encompasses methods for connecting many simple element equations over many small subdomains, named finite elements, to approximate a more complex equation over a larger domain.*
>
> (`https://en.wikipedia.org/wiki/Finite_element_method`)

$\Rightarrow$ mathematical proofs of the FEM
$\Rightarrow$ C++ library (Felisce) implementing the FEM

Motivations

The Finite Element Method (FEM) is the most used method to solve PDEs over meshes.

> *FEM encompasses methods for connecting many simple element equations over many small subdomains, named finite elements, to approximate a more complex equation over a larger domain.*
>
> (`https://en.wikipedia.org/wiki/Finite_element_method`)

$\Rightarrow$ mathematical proofs of the FEM
$\Rightarrow$ C++ library (Felisce) implementing the FEM

Let us machine-check this program!

Motivations

The Finite Element Method (FEM) is the most used method to solve PDEs over meshes.

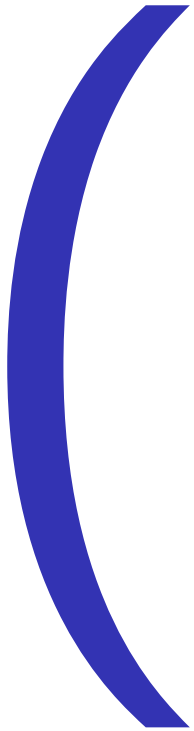
> *FEM encompasses methods for connecting many simple element equations over many small subdomains, named finite elements, to approximate a more complex equation over a larger domain.*
>
> (`https://en.wikipedia.org/wiki/Finite_element_method`)

⇒ mathematical proofs of the FEM
⇒ C++ library (Felisce) implementing the FEM

Let us machine-check this program!

First, let us understand/formally prove the mathematics.

Opening a parenthesis



The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

We developed a Coq library of real analysis that is:

- compatible with the Coq standard library of real numbers,

The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

We developed a Coq library of real analysis that is:

- compatible with the Coq standard library of real numbers,
- with total functions for limit, derivative, integral,

The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

We developed a Coq library of real analysis that is:

- compatible with the Coq standard library of real numbers,
- with total functions for limit, derivative, integral,
- with parametric integrals, two-dimensional differentiability, asymptotic behaviors,

The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

We developed a Coq library of real analysis that is:

- compatible with the Coq standard library of real numbers,
- with total functions for limit, derivative, integral,
- with parametric integrals, two-dimensional differentiability, asymptotic behaviors,
- with a tactic dedicated to derivative proofs.

The Coquelicot library (Boldo, Lelay, Melquiond)

We wanted to prove that

$$u(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right) + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\,d\xi + \frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

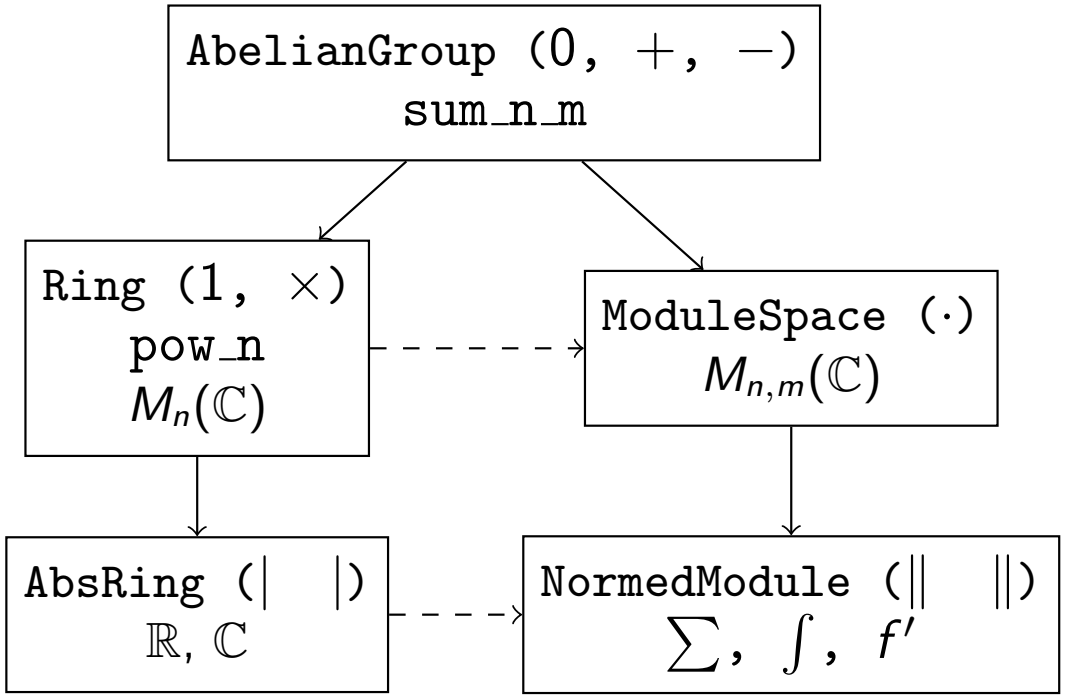
is regular and solution of $\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t)$.

We developed a Coq library of real analysis that is:

- compatible with the Coq standard library of real numbers,
- with total functions for limit, derivative, integral,
- with parametric integrals, two-dimensional differentiability, asymptotic behaviors,
- with a tactic dedicated to derivative proofs.

$\Rightarrow$ Then, it was extended to more than real analysis.

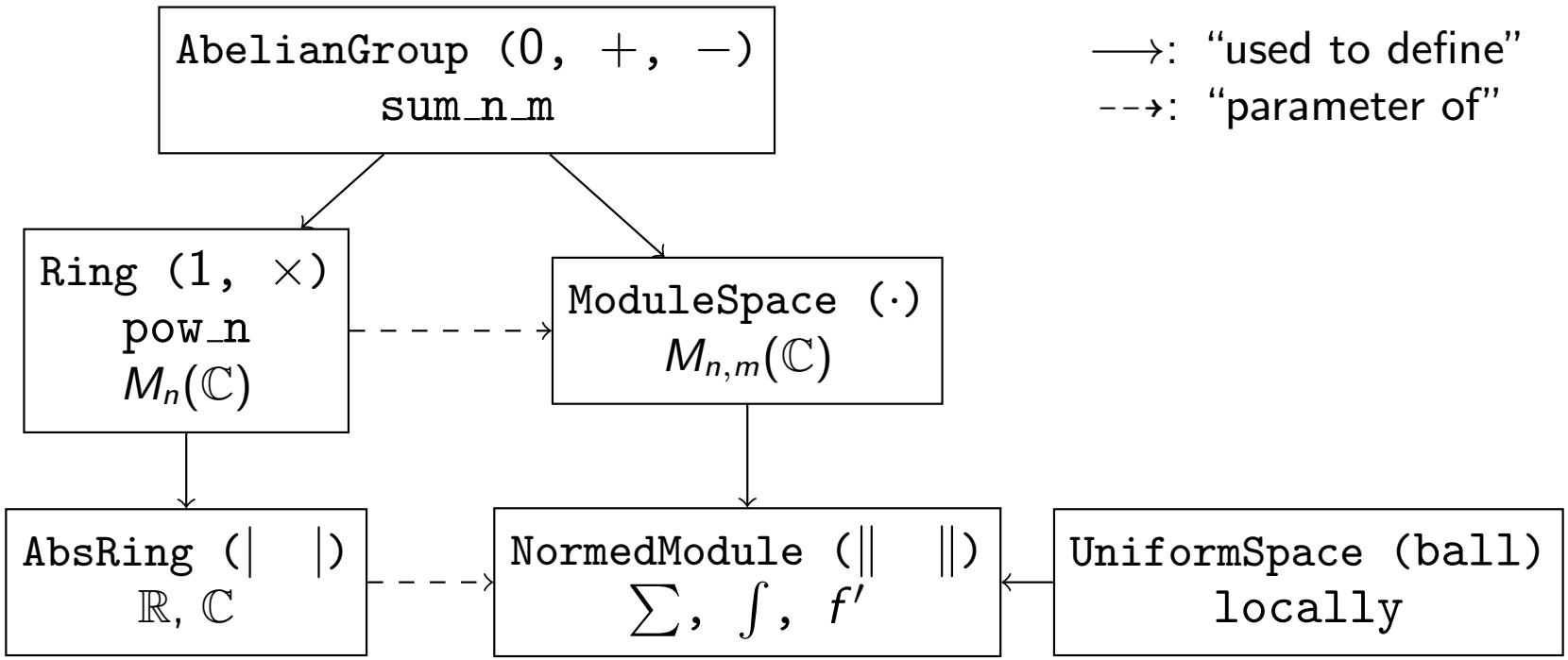
Hierarchy (Lelay, Melquiond)



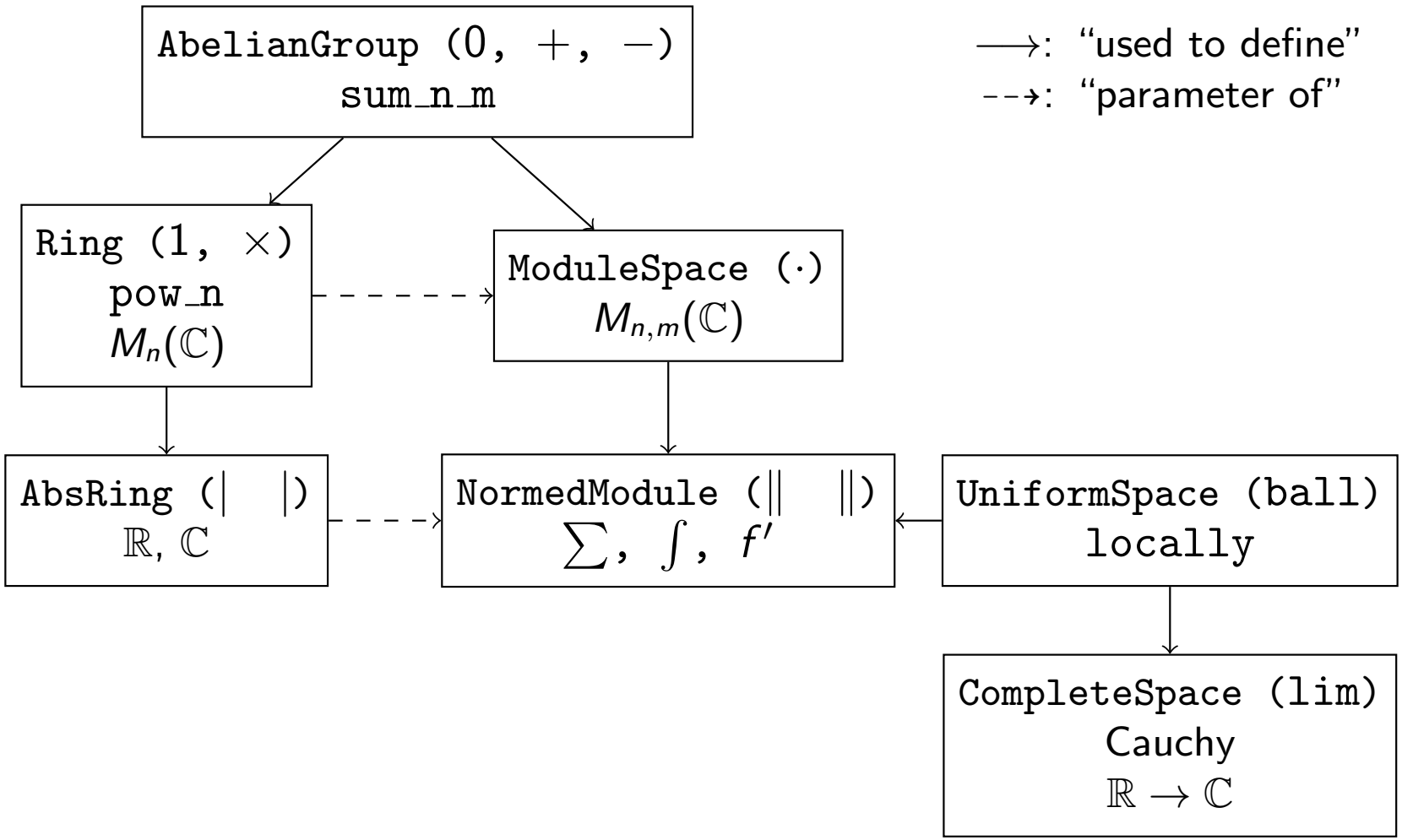
$\longrightarrow$: "used to define"

$\dashrightarrow$: "parameter of"

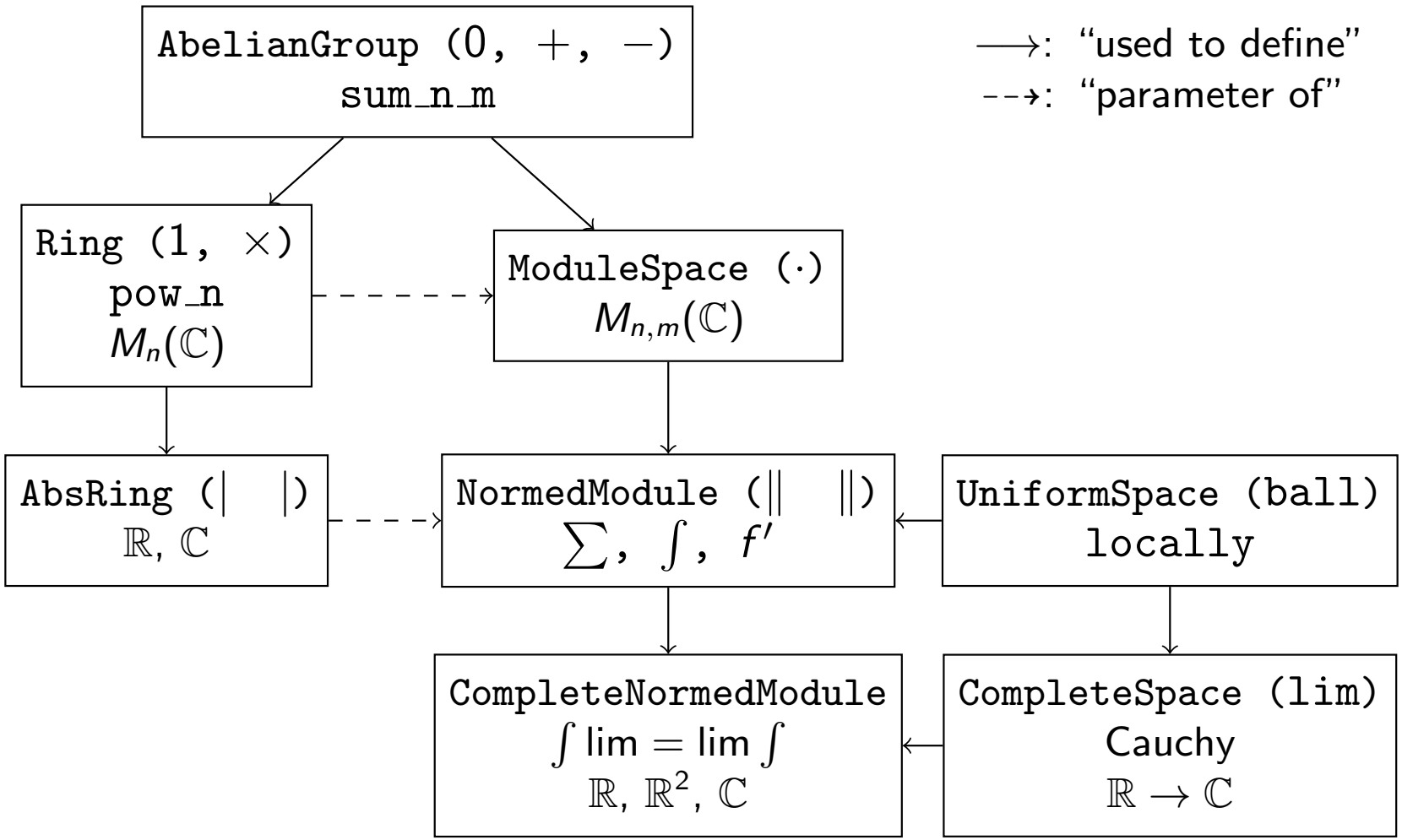
Hierarchy (Lelay, Melquiond)



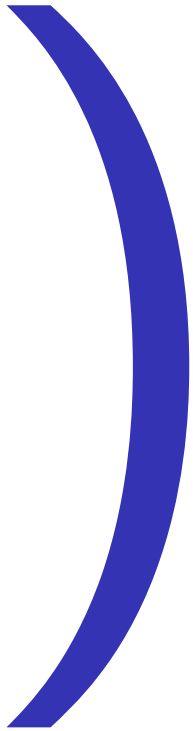
Hierarchy (Lelay, Melquiond)



Hierarchy (Lelay, Melquiond)



Closing a parenthesis



Mathematicians at work

- more 50 pages of mathematical proofs

Mathematicians at work

- more 50 pages of mathematical proofs
- very detailed!

Mathematicians at work

- more 50 pages of mathematical proofs
- very detailed!
- more than 7,000 lines and 220,000 characters

Mathematicians at work

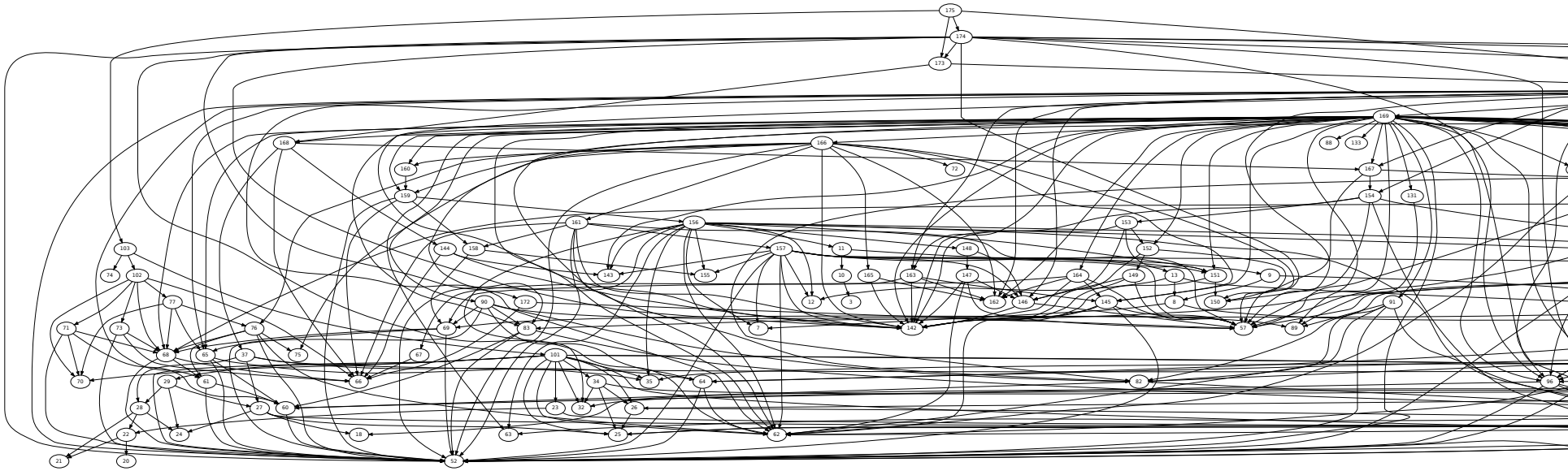
- more 50 pages of mathematical proofs
- very detailed!
- more than 7,000 lines and 220,000 characters
- with dependencies!

Mathematicians at work

- more 50 pages of mathematical proofs
- very detailed!
- more than 7,000 lines and 220,000 characters
- with dependencies!

Mathematicians at work

- more 50 pages of mathematical proofs
- very detailed!
- more than 7,000 lines and 220,000 characters
- with dependencies!



Proof engineering

Let us build upon Coquelicot

+ general spaces

Proof engineering

Let us build upon Coquelicot

+ general spaces
+ many existing theorems

Proof engineering

Let us build upon Coquelicot

+ general spaces
+ many existing theorems
- not always the space we need

Proof engineering

Let us build upon Coquelicot

+ general spaces
+ many existing theorems
- not always the space we need

Proof engineering

Let us build upon Coquelicot

+ general spaces
+ many existing theorems
- not always the space we need

Please note this is still work in progress.

Summary of the work done

- results about functional spaces, linear and bilinear mappings

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$
- 8 equivalences of continuity of linear mappings

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$
- 8 equivalences of continuity of linear mappings
- definitions of pre-Hilbert and Hilbert spaces inside the Coquelicot hierarchy (and first lemmas)

Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$
- 8 equivalences of continuity of linear mappings
- definitions of pre-Hilbert and Hilbert spaces inside the Coquelicot hierarchy (and first lemmas)
- define `clm`: the set of the continuous linear mappings

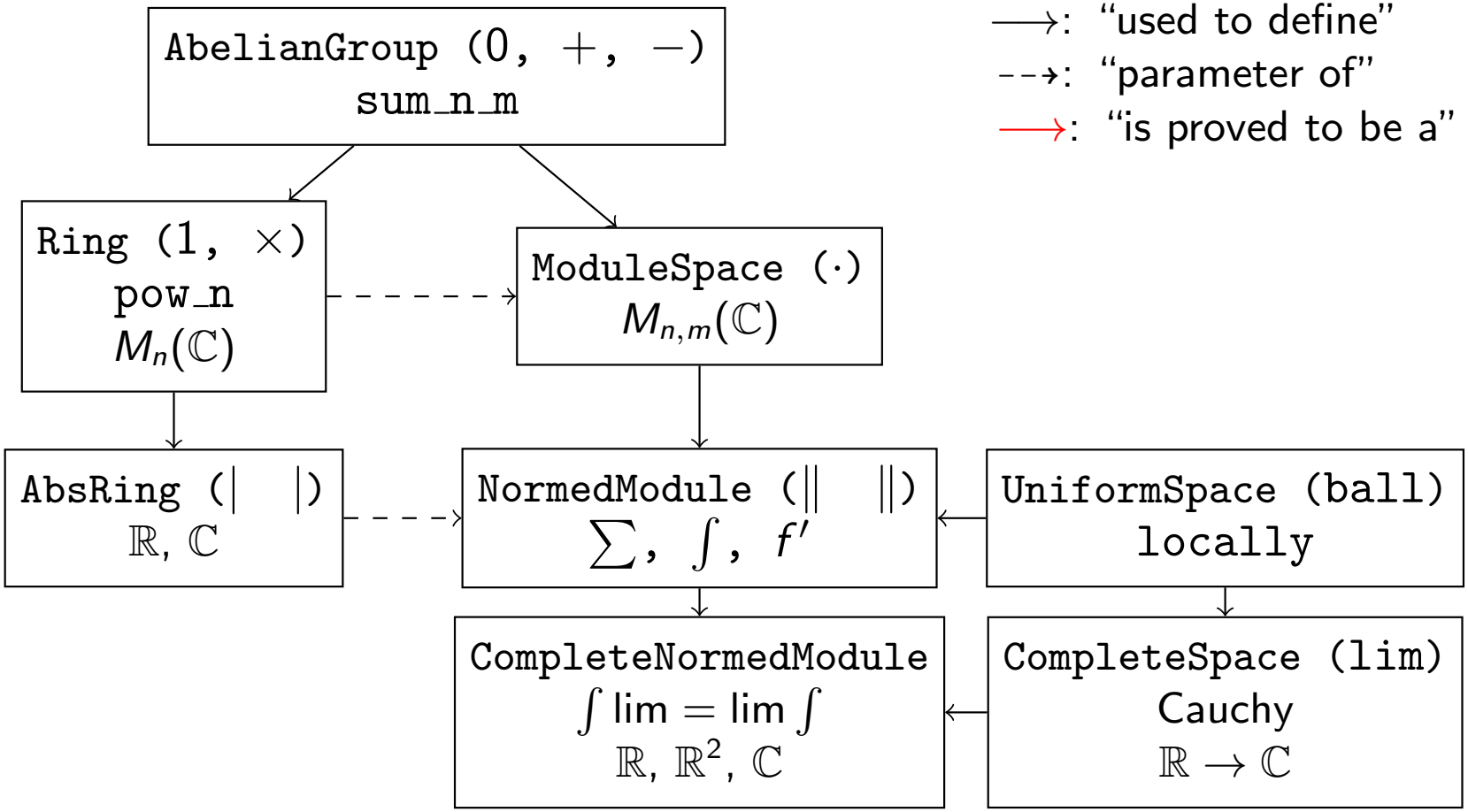
Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$
- 8 equivalences of continuity of linear mappings
- definitions of pre-Hilbert and Hilbert spaces inside the Coquelicot hierarchy (and first lemmas)
- define `clm`: the set of the continuous linear mappings
- prove it is a `NormedModule`, to consider `clm` E (`clm` E $\mathbb{R}$)

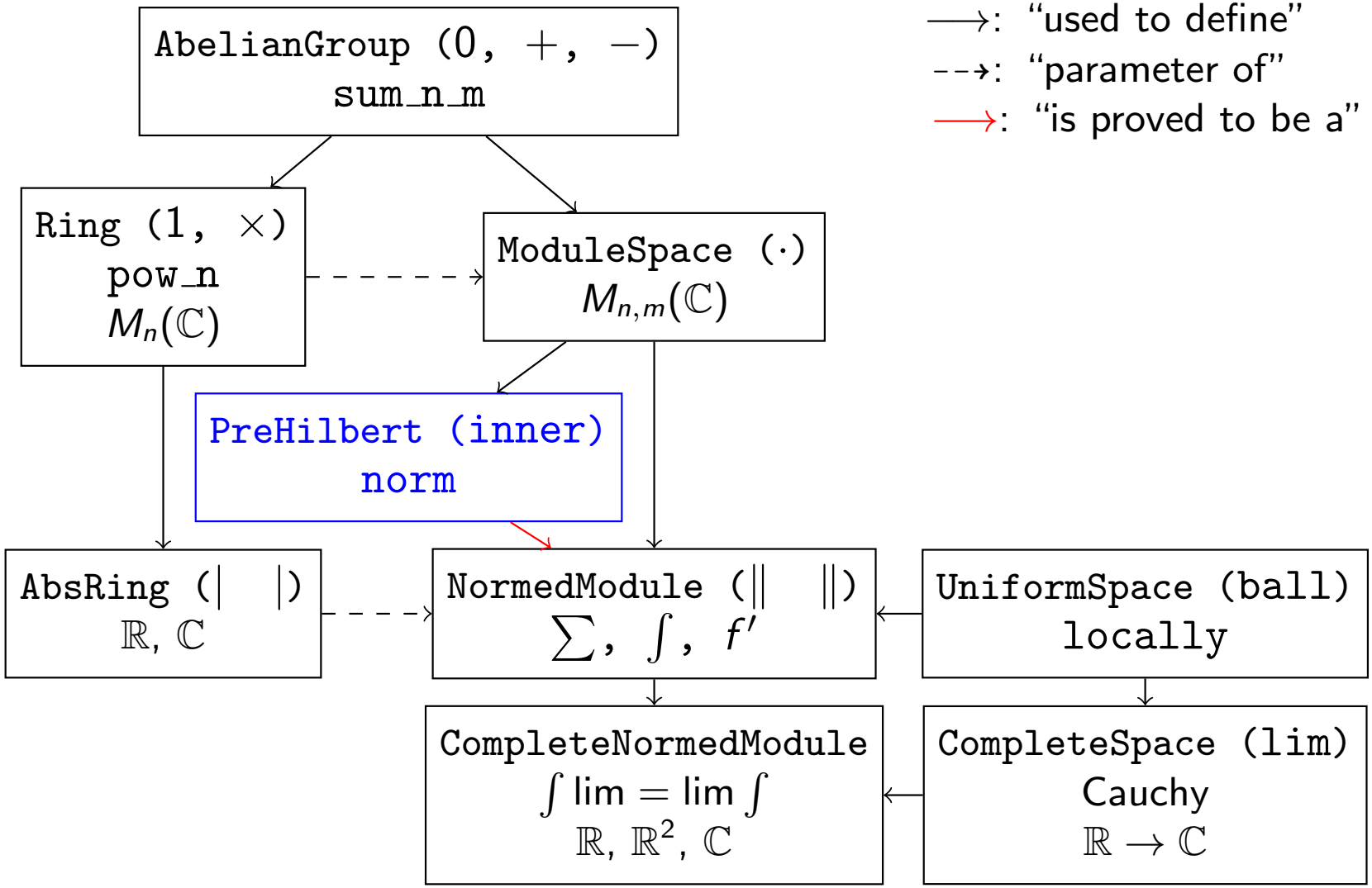
Summary of the work done

- results about functional spaces, linear and bilinear mappings
- fixed-point theorem in a sub-complete normed module
- decide if a space ($\geq$ NormedModule) is only zero
- norm on functions (`operator_norm`) on $\mathbb{R} \cup \{+\infty\}$
- 8 equivalences of continuity of linear mappings
- definitions of pre-Hilbert and Hilbert spaces inside the Coquelicot hierarchy (and first lemmas)
- define `clm`: the set of the continuous linear mappings
- prove it is a `NormedModule`, to consider `clm` E (`clm` E $\mathbb{R}$)
- state Lax-Milgram theorem

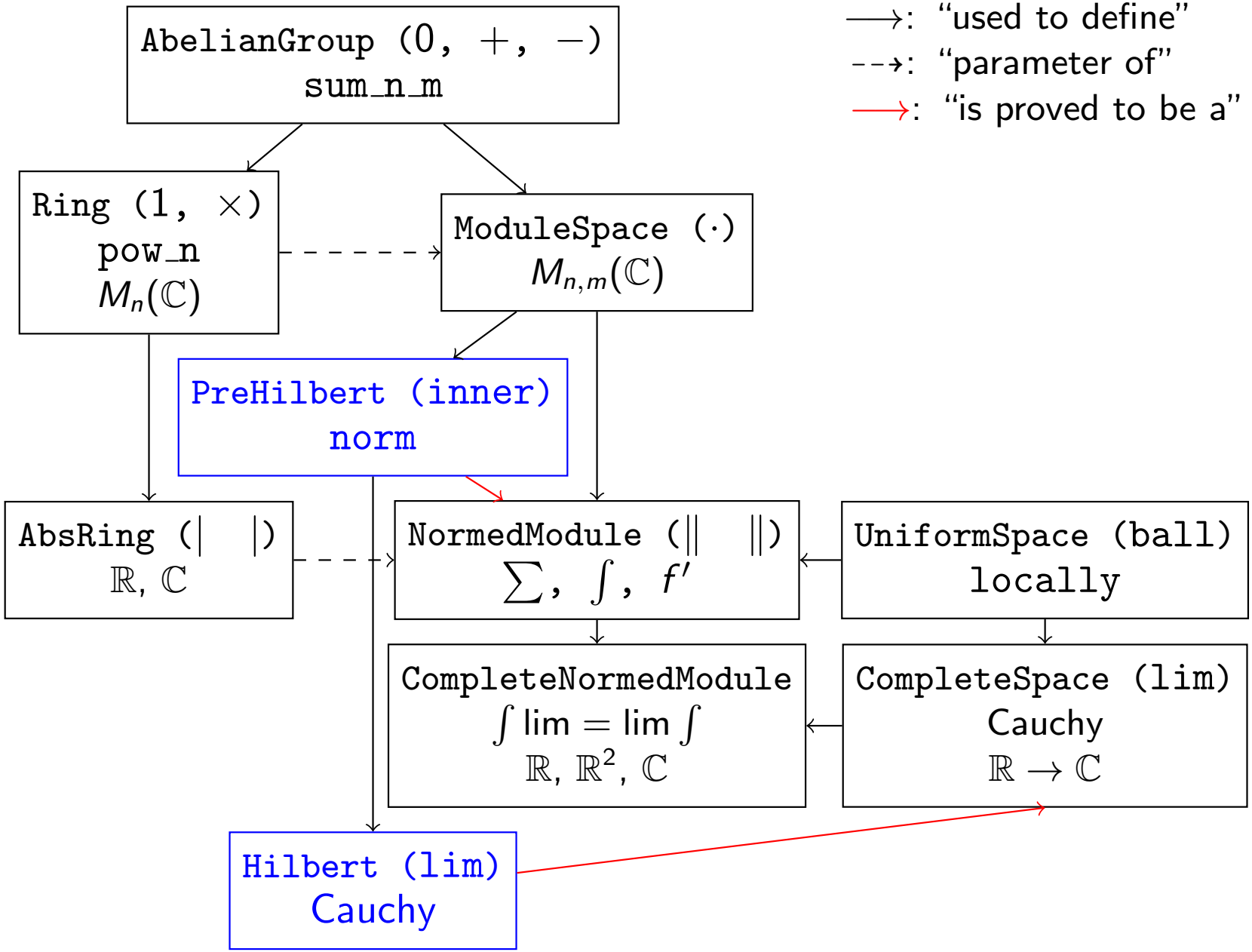
Hierarchy II



Hierarchy II



Hierarchy II



Intermediate conclusion

Problems:

- what is a subgroup?

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\}$

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group
  - overgroup $+ \chi +$ a property of χ
  - ...

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group
  - overgroup $+ \chi +$ a property of χ
  - ...
- canonical structures (both a help and a pain)

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group
  - overgroup $+ \chi +$ a property of χ
  - ...
- canonical structures (both a help and a pain)

Size:

- maths: more than 7,000 lines and 220,000 characters (50 pages)

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group
  - overgroup + χ + a property of χ
  - ...
- canonical structures (both a help and a pain)

Size:

- maths: more than 7,000 lines and 220,000 characters (50 pages)
- Coq: more than 2,300 lines and 56,000 characters (117 lemmas)

Intermediate conclusion

Problems:

- what is a subgroup?
  - dependent type: consider $\{g \in G \mid \chi(g)\} = \{g : G; H : \chi(g)\}$ and prove it is a group
  - overgroup + χ + a property of χ
  - ...
- canonical structures (both a help and a pain)

Size:

- maths: more than 7,000 lines and 220,000 characters (50 pages)
- Coq: more than 2,300 lines and 56,000 characters (117 lemmas)
- still a lot to do!

Outline

Conclusion on the 1-D wave equation discretization

- Very high guarantee

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:
  - all other errors such as pointer dereferencing or division by zero

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:
  - all other errors such as pointer dereferencing or division by zero
  - link with mathematical properties

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:
  - all other errors such as pointer dereferencing or division by zero
  - link with mathematical properties
  - any property can be checked

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:
  - all other errors such as pointer dereferencing or division by zero
  - link with mathematical properties
  - any property can be checked
- expressive annotation language (as expressive as Coq)
  $\Rightarrow$ exactly the specification you want

Conclusion on the 1-D wave equation discretization

- Very high guarantee
- interdisciplinary (formal methods / numerical analysis)
- not only rounding errors:
  - all other errors such as pointer dereferencing or division by zero
  - link with mathematical properties
  - any property can be checked
- expressive annotation language (as expressive as Coq)
  $\Rightarrow$ exactly the specification you want
- an annotated C program to convince numerical analysts

Perspectives

- go deeper into numerical analysis

Perspectives

- go deeper into numerical analysis

$\Rightarrow$ proof of the finite element method

Perspectives

- go deeper into numerical analysis

$\Rightarrow$ proof of the finite element method

$\Rightarrow$ proof of the finite element method library

Perspectives

- go deeper into numerical analysis

$\Rightarrow$ proof of the finite element method

$\Rightarrow$ proof of the finite element method library

$\Rightarrow$ stability (floating-point stability / numerical analysis stability)

Perspectives

- go deeper into numerical analysis

$\Rightarrow$ proof of the finite element method

$\Rightarrow$ proof of the finite element method library

$\Rightarrow$ stability (floating-point stability / numerical analysis stability)

- prove and generalize well-known facts/algorithms/programs from the computer arithmetic community

Perspectives

- go deeper into numerical analysis

$\Rightarrow$ proof of the finite element method

$\Rightarrow$ proof of the finite element method library

$\Rightarrow$ stability (floating-point stability / numerical analysis stability)

- prove and generalize well-known facts/algorithms/programs from the computer arithmetic community

$\Rightarrow$ basic blocks to build upon