Improving Real Analysis in Coq: a User-Friendly Approach to Integrals and Derivatives

Sylvie Boldo, Catherine Lelay, Guillaume Melquiond

Improving Real Analysis in Coq: a User-Friendly Approach to Integrals and Derivatives*

Sylvie Boldo[1], Catherine Lelay[1], and Guillaume Melquiond[1]

INRIA Saclay - Île-de-France, ProVal, Palaiseau, F-91120
LRI, Université Paris-Sud, CNRS, Orsay, F-91405
Sylvie.Boldo@inria.fr
Catherine.Lelay@inria.fr
Guillaume.Melquiond@inria.fr

Abstract. Verification of numerical analysis programs requires dealing with derivatives and integrals. High confidence in this process can be achieved using a formal proof checker, such as Coq. Its standard library provides an axiomatization of real numbers and various lemmas about real analysis, which may be used for this purpose. Unfortunately, its definitions of derivative and integral are unpractical as they are partial functions that demand a proof term. This proof term makes the handling of mathematical formulas cumbersome and does not conform to traditional analysis. Other proof assistants usually do not suffer from this issue; for instance, they may rely on Hilbert's epsilon to get total operators. In this paper, we propose a way to define total operators for derivative and integral without having to extend Coq's standard axiomatization of real numbers. We proved the compatibility of our definitions with the standard library's in order to leverage existing results. We also greatly improved automation for real analysis proofs that use Coq standard definitions. We exercised our approach on lemmas involving iterated partial derivatives and differentiation under the integral sign, that were missing from the formal proof of a numerical program solving the wave equation.

1 Introduction

From Newton and Leibniz during the 17th century, many mathematicians have used integrals and derivatives. Their use is both for pure analysis theorems, but also more recently for applied mathematics. For example, numerical analysis aims at solving ordinary differential equations and partial differential equations. When the solutions are not analytic, it provides algorithms to approximate these solutions and bounds to assert their correctness. Typically, it consists in a numerical schemes over a discrete grid and its convergence proof, meaning that the approximation improves when the grid size decreases.

* This research was supported by the Fœst project (ANR-08-BLAN-0246-01) of the French national research organization (ANR) and by the Coquelicot project of the Digiteo cluster and Île-de-France regional council.

Recent advances in formal proof assistants have shown that they can be applied to various kinds of problems, but analysis and especially numerical analysis was not as much studied as algebra. One reason may be that formalizations of analysis were done years ago and seldom used. This is precisely the case in the standard library of Coq: derivatives and integrals were defined with real numbers a dozen years ago, but the libraries did not evolve with Coq. A more extensive use could have proved the ponderousness of the library. The most blatant example is that derivatives require a proof term to be written. This means that, instead of $f'(x)$ we have to handle (f, x, H) where H is a proof that f is derivable in x. This makes the rewritings clumsy and unpractical. More, this is not the way mathematicians prove their theorems: proofs that functions are regular enough, when present, are side-proofs that do not arise in the main development. Another example is the missing definitions and lemmas about partial derivatives.

As shown in Figure 1, we have extended the standard Coq library with equivalent definitions that are easier to use and with some automations.

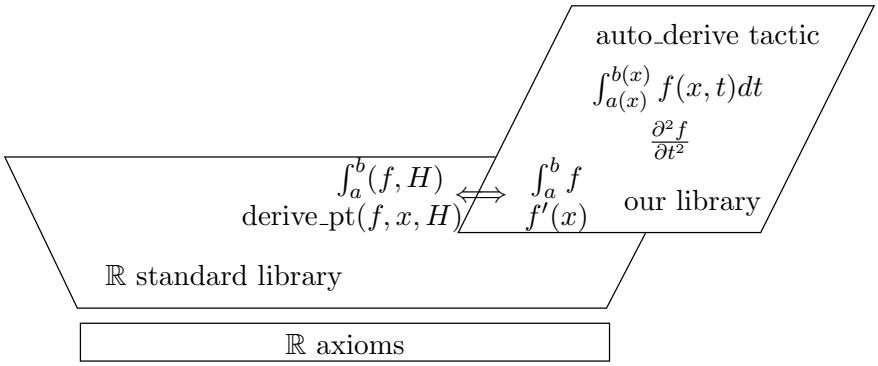

Fig. 1. Sketch of our library: equivalent definitions, additional lemmas and automatic tactic.

To validate this approach, we have applied it to an example coming from a numerical analysis program. We had previously proved the correctness of a program solving the wave equation, but we left out the proofs about the existence and regularity of a solution to the partial differential equation.

The developed library can be downloaded and browsed at:

`http://coquelicot.saclay.inria.fr/results.html`

Section 2 presents the existing formalizations of analysis in proof assistants, and especially in Coq. Section 3 presents the power of the underlying logic of Coq, especially the changes when considering the axiomatic of real numbers. Section 4 presents our design choices for the derivative and the integral. Section 5 presents our application about the wave equation and its required results.

2 State of the Art

We will present the notions of differentiability and integrability as they are defined in various proof assistant such as Coq[1], Isabelle/HOL[2], one of its variant HOL Light, and PVS[3]. We will first present the definitions for being differentiable or integrable, before describing design choices for using derivatives and integrals.

2.1 Differentiability and Integrability

The choices made here are to adopt one of the common mathematical definitions. For differentiability, Coq and PVS use ε-δ-definition directly or through the limit definition:

$$\exists \ell \in \mathbb{R},\ \forall \varepsilon > 0,\ \exists \delta > 0,\ \forall h \in \mathbb{R},\ (h \neq 0 \wedge |h| < \delta) \Rightarrow \left| \frac{f(x+h) - f(x)}{h} - \ell \right| < \varepsilon$$

In Isabelle/HOL, the same limit of the Newton's difference quotient is used, but limit is a more generic notion as it is defined in a topological space based on a field. Another approach is implemented in additional libraries: the Frechet derivative in a real normed vector space. In C-CoRN, a constructive formalization of real numbers for Coq, the previous definition is modified to get a uniform differentiability [5] on a close interval $[a; b]$, *i.e.* there is a single δ for all $x \in [a; b]$.

To define Riemann integrability, Coq defines the integral on step functions and then uses the traditional ε-δ-definition:

$$\begin{array}{c} \forall \varepsilon > 0, \text{ there are two step functions } \varphi, \phi : [a; b] \to \mathbb{R}, \text{ such as} \\ (\forall t \in [a; b],\ |f(t) - \varphi(t)| \leq \psi(t)) \wedge |\int \psi| < \varepsilon \end{array}$$

The value of the integral is then defined as the limit of $\int \varphi$ when $\varepsilon \to 0$.

PVS defines Riemann integrability as the convergence of Riemann sums [4]. Both definitions are mathematically equivalent. A difference is that the PVS definition gives the value of the integral, while Coq definitions only provides the step functions φ that has to be integrated to get the value.

C-CoRN uses another equivalent definition of Riemann integrability: the convergence of Darboux sequences. As for Riemann sums in PVS, the integral value is directly obtained from the definition.

Isabelle/HOL does not define Riemann integrals, but both Gauge and Lebesgue integrals, which are more general but less intuitive.

In a former library about reals in Isabelle/HOL [7], and in ACL2(r) [8], differentiability was defined from non-standard analysis where the formal notion

[1] `http://coq.inria.fr/stdlib/index.html`

[2] `http://www.cl.cam.ac.uk/research/hvg/isabelle/dist/library/HOL/index.html`

[3] `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/analysis-details.html`

of "infinitely close", *i.e.* difference is less than all positive real numbers, replaces the informal notion of "sufficiently close" corresponding to a common formula stating that $\forall \varepsilon > 0, \ldots$, the difference may be made smaller than ε.

2.2 Derivative and Integral

In pen-and-paper mathematics, we understand $f'(x)$ as the function f is differentiable at x and $f'(x)$ is its derivative value. But the corresponding definitions and their uses in formal proof assistant are not straightforward and differ, among others as the underlying logic is different.

PVS is the closest to the mathematical point of view by using Type Correctness Conditions (TCCs). A user may write a statement containing occurrences of $f'(x)$ without justification and PVS generates additional goals to prove that f is actually differentiable in x. PVS tries to infer automatically these additional goals from the context. The same approach is used in ALC2(r) [9].

Isabelle/HOL uses its Hilbert's epsilon applied to the differentiability property. The user can then write $f'(x)$ without proof, but has to prove that f is differentiable at x before using differentiation rules.

In Coq, the derivative is a partial function taking explicitly a proof term of differentiability. As a derivative cannot be defined or used without this proof term, this is cumbersome to use. One of the goals of the MathClasses project [12] is to provide assistance work with these proof terms by trying to infer them.

3 Logical Foundations

3.1 An Overview of Coq's Logic

The formal system of Coq is an intuitionistic logic called the Calculus of Inductive Constructions. A salient point is that, whenever one wants to prove a property of the form $\exists x,\ P(x)$, one has to actually build a witness x such that $P(x)$ holds. This is different from classical logic where one could have simply proved $\neg\forall x,\ \neg P(x)$ and be done.

Another peculiarity of Coq's logic is related to its type hierarchy. Logical formulas have type `Prop` while values and functions have their types in `Type`. The point of interest is that `Prop` is *noninformative*, that is, one can only use the witness of an existential property $\exists x,\ P(x)$ inside the proof of logical formula. A witness can never be used inside a logical formula itself, or more importantly to define a value or a function. For instance, being able to prove a formula $\forall x : X,\ \exists y : Y,\ P(x, y)$ does not provide a function $f : X \to Y$ such that $\forall x,\ P(x, f(x))$. Note that strengthening the property so that y exists and is unique does not help either.

The traditional way in Coq to circumvent this issue is in the use of *specification* types. They are existential types denoted $\{x : X \mid P(x)\}$ and they contain dependent pairs (x, p) such that x is an element of type X and p is a proof of the logical formula $P(x)$. The upside is that witnesses are readily available for

use inside values and functions. The downside is that these types do not live in `Prop` and are therefore less natural to manipulate.

For instance, consider the predicate `derivable_pt` from the standard library that states that a function f is differentiable at point x. It is in fact a notation for the specification type that associates a value ℓ with the proof that ℓ is the limit of the the slope function of f at point x. As a consequence, knowing that a function is differentiable gives access to the value of its derivative. But it also means that trying to express that both f and g are functions that are differentiable as $\texttt{derivable\_pt}(f, x) \wedge \texttt{derivable\_pt}(g, x)$ will be rejected by Coq. Indeed, this formula is ill-typed since neither members are logical formulas.

In the same way that one can extract a witness from the formula $\exists x,\ P(x)$ only when performing a proof, the information about the disjunction $P \vee Q$ cannot be used to make a choice inside a value or a function. Again, types outside `Prop` have been introduced to offer this possibility ; they are denoted $\{P\} + \{Q\}$ in Coq and their values can be used to select the branch of an if-then-else.

To complete this overview, one should mention that subsets of a type T are usually represented by predicates, that is, functions of type $T \to \texttt{Prop}$. As a consequence, we will indifferently note the containment property by the logical formulas $x \in S$ or $S(x)$.

3.2 Coq's Standard Axioms for Real Numbers

The formalization of real numbers from the standard library is axiomatic rather than constructive. Instead of building reals as Cauchy sequences or Dedekind cuts of rational numbers and proving their properties, Coq developers have chosen to assume the existence of a set with the usual properties of the real line. In other words, the standard library states that there is a set `R`, some arithmetic operators $-$, $+$, $\times$, $\square^{-1}$, and a comparison operator $<$, that have the properties of an ordered field.

The previous axioms are not controversial. Below are the three axioms that state that `R` is Archimedean, closed under the supremum bound, and its order is decidable:

- `archimed`: There is a function $\texttt{up} : \texttt{R} \to \texttt{Z}$ such that $\forall x \in \texttt{R},\ x < \texttt{up}(x) \leq x + 1$.
- `completeness`: As long as one can prove that a subset E of `R` is not empty ($\exists x,\ E(x)$) and is bounded ($\exists M,\ \forall x,\ E(x) \Rightarrow x \leq M$), one gets a value of type $\{y : \texttt{R} \mid y \text{ is an upper bound of } E \text{ and it is the least one}\}$.
- `total_order_T`: Given two real numbers x and y, there is a value of type $\{x < y\} + \{x = y\} + \{x > y\}$.

While all three axioms could have been defined as logical formulas in `Prop`, they were not. In other words, it is equivalent to having three functions that can compute the integer part of a real number, the supremum of a bounded subset of `R`, and the order of two numbers. Notice that excluded-middle is not one of the axioms of Coq's standard real numbers. While the standard library sometimes

imports this axiom, the CoqTail project[4] has shown that it was often unneeded. So we restrict our use of classical reasoning to goals that are the negation of a logical formula.

Notice also that the `completeness` axiom has an intuitionistic feel to it. Indeed, applying the axiom does not allow to extract any proof witness other than the one that had to be created beforehand to prove that E is not empty. At best, one can derive the following property which contains a double negation:

$$\forall \varepsilon > 0,\ \neg\neg\exists x,\ y - \varepsilon \leq x \leq y \wedge E(x).$$

3.3 Small Omniscience Principle

Let P be a decidable predicate on natural numbers. The small omniscience principle, also known as Markov's principle, states that one can decide whether the property never holds, and if it does, return n such that $P(n)$ holds. In Coq syntax, the principle is stated $\{n \mid P(n)\} + \{\forall n,\ \neg P(n)\}$. It cannot be derived without axioms in Coq, as it would require the ability to test all the values of n at once. Thanks to the axioms on real numbers, it becomes possible.

The way we have proved Markov's principle is as follows. Since P is decidable, we can build a function $f(n)$ that returns 1 if $P(n)$ holds and 0 otherwise. Let us consider the subset of real numbers $\{f(n) \times 2^{-n} \mid n \in \mathbb{N}\}$. It is nonempty and bounded by 1, thus it has a supremum (`completeness`). This supremum can be tested against 0 (`total_order_T`). If it is zero, we deduce $\forall n,\ \neg P(n)$. Otherwise we compute its discrete logarithm (`archimed`) which will act as a witness for building a value of type $\{n \mid P(n)\}$.

3.4 Bounds and Limits

Now that we have proved this principle, we can use it to decide whether a subset E of real numbers is bounded, which is a precondition for computing its supremum. First, let us consider the family of subsets $E_n = \{0\} \cup (E \cap (-\infty; n])$. They are nonempty and bounded, so they have a supremum s_n. As a consequence, deciding whether E is bounded and computing an upper bound is a matter of applying Markov's principle to decide the following alternative:

$$\{M \mid \forall n,\ s_n \leq M\} + \{\forall M,\ \neg(\forall n,\ s_n \leq M)\}.$$

This is Lemma `Rbar_ub_dec` of our development. Its proof requires that $\forall n,\ s_n \leq M$ is decidable, which is just a consequence of Markov's principle applied to the decidable predicate $n \mapsto M < s_n$.

Let us define the set `Rbar` $= \mathtt{R} \cup \{-\infty, +\infty\}$. Since we are able to decide whether a set is bounded, we can define supremum and infimum functions for nonempty subset of `Rbar`. Let us consider a sequence $(u_n)_{n\in\mathbb{N}}$ of elements of `R`

[4] `http://coqtail.sourceforge.net/`

(or `Rbar`). The set of its values is nonempty, thus we can define its superior and inferior limits:

$$\limsup_{n\in\mathbb{N}} u_n = \inf_{m\in\mathbb{N}} (\sup_{n\in\mathbb{N}} u_{m+n}) \qquad \liminf_{n\in\mathbb{N}} u_n = \sup_{m\in\mathbb{N}} (\inf_{n\in\mathbb{N}} u_{m+n})$$

At this point, we arbitrarily define a function `Lim_seq`, which takes a sequence $(u_n)_n$ of real numbers and returns the real $(\limsup u_n + \liminf u_n)/2$. If $(u_n)_n$ is a converging sequence, `Lim_seq`(u_n) is the actual limit of the sequence, since inferior, superior, and plain limits are then equal. Note that this also gives us a way to decide whether an arbitrary sequence is converging: we just have to compare its inferior and superior limits. There are simpler possibilities for defining `Lim_seq`, but this one offers the equality `Lim_seq`$(\alpha \cdot u_n) = \alpha \cdot$ `Lim_seq`(u_n) for any real α without requiring $(u_n)_n$ to converge.

Therefore, we now have an operator able to compute the limit of any converging sequence and otherwise return an undefined value (as far as the user is concerned). We can similarly define the limit of a function f in a point x by

```
Definition Lim (f : R → R) (x : R) :=
  Lim_seq (fun n ⇒ f (x + /INR n)).
```

Again, if the function has a limit at this point, operator `Lim` returns it, since $\lim_{u\to x} f(u)$ is then equal to $\lim_{n\to\infty} f(x + 1/n)$. Otherwise it returns an undefined real number.

3.5 Compactness

Limits are the basis for doing real analysis. Another important tool is the property of compactness, which has numerous applications in traditional mathematics. For instance, a function continuous on a compact is uniformly continuous. Unfortunately, the compactness property is inherently classical, up to the point that constructive mathematics tend to redefine continuity so that it actually means uniform continuity in order to avoid compactness [6]. Our goal is to stay as close as possible from traditional analysis, so dropping compactness is not a solution.

One of the definitions of a compact set is a set such that, from any cover with open sets, one can extract a finite subcover. Yet in most of the proofs we are interested in, we do not need the whole power of this property. Indeed, the extracted sets are useless, only their minimum diameter matters. Moreover, the finiteness property is only useful so that this minimum is nonzero. As a consequence, we can substitute to the traditional definition a property of interval $[a, b]$ related to Cousin covers and gauge functions:

$$\forall \delta : \mathbb{R} \to \mathbb{R}^{+*},\ \{\delta' : \mathbb{R}^{+*} \mid \forall x \in [a,b],\ \neg\neg\exists t \in [a,b],\ |x-t| < \delta(t) \wedge \delta' \leq \delta(t)\}.$$

The proof of this lemma (`compactness_value`) requires us to construct a value δ' that satisfies the property. It is defined by the formula

$$\delta' = \sup\ \{d \mid d \leq 1 \wedge \forall x \in [a,b],\ \exists t,\ |x-t| < \delta(t) \wedge d \leq \delta(t)\}.$$

Thus the only thing we have to prove is that this supremum is not equal to zero. The idea behind our proof is the one found in most cover-based proofs: consider the widest interval $[a, b')$ such that the property holds (b' can be expressed as a supremum value) and prove that $b' = b$. As mentioned before, the completeness axiom does not provide us with proofs that the property holds before the supremum, only that it cannot be false, hence the double negation in the statement of the lemma.

We have proved the compactness property not just for segments of the real line but for any n-orthotope.

4 Derivatives and Integrals

4.1 Derivative

From the previous definition of a limit function in Section 3.4, we defined a total derivative function:

$$\texttt{Definition Derive}(f, x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

and we proved that if the f is differentiable at x, the result of our function is equal to the derivative from the standard library.

As explained before, this definition allows us to state derivative without a proof term. For example, to prove the differentiability and the value of the derivative of a sum of differentiable functions in Coq, we previously had to write in two goals:

```
Lemma derivable_pt_plus : forall f g x,
  derivable_pt f x → derivable_pt g x
    → derivable_pt (fun t ⇒ f t + g t) x.
Goal forall f g x (prf : derivable_pt f x) (prg : derivable_pt g x),
  derive_pt (fun t ⇒ f t + g t) x (derivable_pt_plus f g x prf prg)
    = derive_pt f x prf + derive_pt g x prg.
```

With our function `Derive`, we write a more readable statement without any dependent type:

```
Goal forall x, ex_derive f x → ex_derive g x →
  ex_derive (fun t ⇒ f t + g t) x
  ∧ Derive (fun t ⇒ f t + g t) x = Derive f x + Derive g x.
```

Moreover, properties on the limit allow us to get theorems with few or no precondition, that lightens the proof burden. For example, the fact that "$(a \cdot f)' = a \cdot f'$" has no precondition: we do not have to prove that f is differentiable. This is impossible when using a derivative defined on top of Hilbert's epsilon.

4.2 Riemann and Darboux integrals

We chose the Darboux integral and defined our integral number as the limit of:

$$\texttt{RInt\_val}\,(f,a,b,n) = \frac{b-a}{2^n}\sum_{k=0}^{2^n-1} f\left(\frac{x_k+x_{k+1}}{2}\right)$$

where $\forall k \in [\![0;2^n]\!],\ x_k = a + k\cdot(b-a)/2^n$.

We proved the equivalence with Coq's Riemann integral. For that, we carefully chose some Darboux sequences, proved their equivalence to Riemann sums and deduced their convergence.

The Riemann integrability is based on step functions. Unfortunately, step functions from the standard library are hard to build and to use: a step function is built from a function f and two lists lx and ly which must satisfy five conditions. These conditions are difficult both to prove and to use. For example, the last one states that $\forall i < |lx| - 1,\ \forall x \in (lx_i; lx_{i+1}),\ f(x) = ly_i$. This is unpractical as it does not provide any information about the values $f(lx_i)$.

We then chose to define new step functions based on ssreflect sequences [10]:

```
Record SF_seq {T : Type} := mkSF_seq {SF_h : R ; SF_t : seq (R * T)}.
```

Using ssreflect libraries, our step functions were easier to define and use. For example, to define the step function needed for $\texttt{RInt\_val}\,(f,a,b,n)$, we use:

```
Definition SF_val_seq (f : R → R) (a b : R) (n : nat) : SF_seq :=
  SF_seq_f2 (fun x y ⇒ f ((x+y)/2)) (RInt_part a b n) 0.
```

where `RInt_part a b n` is the partition used in this proof.

We then define the following Darboux sequences

$$\texttt{RInt\_sup}\,(f,a,b,n) = \frac{1}{2^n}\sum_{k=0}^{2^n-1}\left(\sup_{[x_k;x_{k+1}]} f\right)\chi_{[x_k;x_{k+1}]}$$

where $\chi_{[x_k;x_{k+1}]}$ is the characteristic function of $[x_k; x_{k+1}]$. We also define in the same way $\texttt{RInt\_inf}\,(f,a,b,n)$ with a inf. At last, we define the integrability `ex_RInt` as the convergence of both Darboux sequences to the same limit in $\mathbb{R}$.

A first interesting point is that our choice for $(x_k)_{k\in[\![0;2^n]\!]}$ make these sequences monotonic. We can then replace Darboux sequences' limit with a supremum for `RInt_inf` and an infimum for `RInt_sup` which are total function and require lighter conditions than a limit. We easily deduce the the fact that f is bounded, needed to have $\texttt{RInt\_sup}\,(f,a,b,n)$ and $\texttt{RInt\_inf}\,(f,a,b,n)$ as real numbers, from theorem hypotheses.

Another interesting point is that the implication from Riemann integrability to Darboux sequences uses an unusual convergence of Riemann sums :

$$\exists \ell \in \mathbb{R},\ \forall \varepsilon > 0,\ \exists \delta > 0,\ \forall(\sigma,\xi),\ \max_{0\le k\le n} |\sigma_{k+1} - \sigma_k| < \delta \Rightarrow |S(f,\sigma,\xi) - \ell| < \varepsilon$$

where $S(f,\sigma,\xi) = \sum_{k=0}^{n} f(\xi_k)\cdot(\sigma_{k+1}-\sigma_k)$ is a Riemann sum, n is length of ξ, $n+1$ the length of σ, $\sigma_0 = a$, $\sigma_{n+1} = b$ and $\forall k \le n$, $\sigma_k \le \xi_k \le \sigma_{k+1}$. The convergence is based here on a subset of finite sequences instead of usual real numbers.

5 Application

5.1 Case study and d'Alembert's Formula

Our main application is part of a project aiming at proving numerical analysis programs. The case study was a C program that implements a numerical scheme for the resolution of the one-dimensional acoustic wave equation. This corresponds to the oscillation of an attached rope where c is the constant propagation velocity, which depends on the section and density of the string. More precisely, we consider the following initial-boundary value problem: we have the initial values u_0 and u_1, a source term s and we want to compute an approximation of the exact solution u of

$$\begin{aligned}
\forall t \geq 0,\ \forall x \in [x_{\min}, x_{\max}],\ & \frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = s(x,t),\\
\forall x \in [x_{\min}, x_{\max}],\ & \frac{\partial u}{\partial t}(x,0) = u_1(x),\\
\forall x \in [x_{\min}, x_{\max}],\ & u(x,0) = u_0(x),\\
\forall t \geq 0,\ & u(x_{\min},t) = u(x_{\max},t) = 0
\end{aligned}$$

To actually compute an approximation of the solution u, we chose the second order centered finite difference scheme, also known as three-point scheme. The size of the grid is $(\Delta x, \Delta t)$ and the value $u_j^k \approx u(j\Delta x, k\Delta t)$ is given by

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

and similar formulas that depend on u_0 and u_1 for the initializations u_j^0 and u_j^1.

We proved that rounding errors do not endanger the results of the numerical scheme [1]. We also formalized in Coq the numerical scheme and proved its convergence, for an infinite rope [2]. In that work, the differentiation operator was an arbitrary function. We did not define it nor did we assume any of its properties. We only used the fact that u is a solution of the partial differential equation expressed using this operator. This fully corresponds to the way mathematical proofs are done: we put $f'(x)$ or $\frac{\partial^2 u}{\partial t^2}(x,t)$ and not $\mathrm{diff}(f,x,H)$ where H is a proof that f is derivable in x or $\mathrm{diff}(z \to \mathrm{diff}(y \to u(x,y), z, H_1), x, H_2)$ where H_1 and H_2 are adequate proof terms. We also needed the regularity of this solution: it is supposed to be near its Taylor expansion with the usual mathematical bounds (see below).

Later, we proved the full C program [3]. As we needed to precisely specify what the program was supposed to compute, we defined each derivative as the limit of $\frac{f(x+h)-f(x)}{h}$ when h goes to zero. We used the Frama-C platform with the Jessie plugin and the specification of the C program is described in C comments called annotations. As the language of these annotations is first-order logic, we could not define in our specifications a differentiation operator, but had to define each of the four derivatives as a limit (with a $\forall\varepsilon, \exists\delta \ldots$ formula). This

fully specifies the derivatives but was unpractical and difficult to read. Yet, this meant (at last) a link between our previous work and real derivatives. But as our previous work required a differentiation operator, we had to actually provide it. The chosen solution was to define it as a parameter and add an axiom stating that, if the function is differentiable, the result of this operator is the expected derivative. This axiom, similar to a Hilbert ε operator, was not satisfactory. Thanks to the formalization presented in this paper, we can now create this operator as a function and we can now get rid of that axiom.

The other axioms needed by this development are the following ones: the existence of a solution to the partial differential equation and its regularity.

About the existence, the mathematical proof is simpler than expected as this equation has an analytical solution. More precisely, the following d'Alembert's formula

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

defines a function that is solution to the previous partial differential equation. We define α, β, and γ, as parts of this formula that will be used below. Just note that they are of increasing difficulty to handle and derive.

5.2 Taylor expansions

The regularity of the solution u is the base of the convergence of the chosen numerical scheme. In the scheme statement, since the grid sizes are small, we can recognize discrete derivatives:

$$\frac{u_j^k - u_j^{k-1}}{\Delta t} \approx \frac{\partial u}{\partial t}(j\Delta x, k\Delta t) \qquad \frac{u_j^k - 2u_j^{k-1} + u_j^{k-2}}{\Delta t^2} \approx \frac{\partial^2 u}{\partial t^2}(j\Delta x, k\Delta t)$$

and similarly for space derivatives. The discrete equation is the exact discrete analog of the continuous wave equation.

Our definition of the Taylor polynomial is the usual one:

$$\mathrm{TP}_n(f,x,t) = \sum_{p=0}^{n} \frac{1}{p!}\left(\sum_{m=0}^{p} \binom{p}{m} \cdot \frac{\partial^p f}{\partial x^m \partial t^{p-m}}(x,t) \cdot \Delta x^m \cdot \Delta t^{p-m}\right)$$

For the main iteration, we need to guarantee that the difference between the function and its order-4 Taylor polynomial is proportional to $(\sqrt{\Delta x^2 + \Delta t^2})^4$. For the initializations, we also need this property at level 3.

We first proved the common Taylor-Lagrange theorem and extended it to its two-dimensional version we really needed here. We had to prove the Schwarz theorem to be able to switch derivatives in space and time. Note that the hypotheses required to make this switch possible are strong (existence and continuity of both second-order derivatives).

5.3 Automation

As explained, the two unproved properties from the original development are that the solution exists and is sufficiently regular. Existence has already been tackled thanks to a reflexive Coq tactic for proving differentiability [11]. The tactic was limited: it could handle expressions with one variable only. As a consequence, while it could automatically perform differentiability proofs on the α and β parts of d'Alembert's formula, human intervention was needed for differentiating under the integral sign of γ. The reason for this shortcoming was the need for proof terms in derivatives and integrals. For instance, the term $\int_{a(x)}^{b(x)} \frac{\partial f}{\partial x}(x,t)dt$ contains a proof that $\frac{\partial f}{\partial x}(x,t)$ can be integrated for t between $a(x)$ and $b(x)$, while the term $\frac{\partial f}{\partial x}(x,t)$ itself contains a proof that f has a first derivative at any point (x,t) of a domain that depends on x. This nesting of values and proof terms ended up being out of reach of our tactic.

For the existence, we only had to consider four partial derivatives of γ. So, despite the absence of automation, we succeeded in formally proving it in Coq. For this work, however, we wanted to prove not only the existence but also the regularity, which means manipulating tens of partial derivatives of γ. This makes it out of reach of a non-automated proof. So we have improved the original tactic now that we have got rid of proof terms in values. The new tactic `auto_derive` still produces side conditions, but the values no longer depend on their proofs, which means the tactic is now able to differentiate below the integral sign.

The tactic is meant to help proving statements of the form

```
derivable_pt_lim f x l
```

that is, f has a derivative at point x and it is equal to l. A variant of the tactic is able to tackle goal where l is not yet known. The tactic first performs a reification of the function f into an inductive object describing the expression. Variables are encoded using De Bruijn's indexes. For instance, the inductive object for

$$y \mapsto \int_0^{2y} (g(y) + z)dz$$

is

```
(Int                                        (* integral *)
  (Binary Eplus                             (* addition *)
    (AppExt 1 g [:: Var 1])      (* application: g(y) *)
    (Var 0))                                    (* z *)
  (Cst 0)               (* lower bound of integral: 0 *)
  (Binary Emult (Cst 2) (Var 0))) (* upper bound: 2y *)
```

Operator `D` is then applied to this inductive object. This is a recursive function that differentiates an expression and generates side conditions. For instance, given an object representing $x \mapsto 2 \cdot f(x)$, it produces an object representing $x \mapsto 2 \cdot f'(x)$ and a side condition that f can be differentiated at the considered point. Lemma `D_correct` states that the generated object is the actual derivative when the side conditions hold. The tactic simply applies this lemma to the current goal, thus solving it, assuming the user can prove the side conditions.

Consider the following script that proves that $\frac{\partial^2 \alpha}{\partial x^2}$ exists and is equal to α_{20}.

```
Definition alpha x t := 1/2 * (u0 (x + c*t) + u0 (x - c*t)).
Definition alpha20 x t :=
  1/2 * (Derive_n u0 2 (x + c*t) + Derive_n u0 2 (x - c*t)).

Lemma alpha_20_lim : forall x t,
  is_derive_n (fun u ⇒ alpha u t) 2 x (alpha20 x t).
Proof.
intros x t. unfold alpha.
auto_derive_2.
```

The `auto_derive_2` is just an ad-hoc tactic developed for this example. It simply applies `auto_derive` twice in a row, so as to prove properties on iterated derivatives. After executing the tactic, the user is left with several goals to prove. They state that function u_0 can be differentiated with respect to the first variable at points $(v + c \cdot t)$ and $(v - c \cdot t)$ for an arbitrary real v around x. They also state that the first partial derivative of u_0 can be differentiated with respect to the first variable at points $(x + c \cdot t)$ and $(x - c \cdot t)$. Finally, the last goal the user has to prove is the equality between $\alpha_{20}(x, t)$ and the expression obtained by automatic differentiation, which is straightforward.

6 Conclusion

We have presented a Coq development for real analysis that aims at being closer to the traditional way of handling analysis theorems in pen-and-paper proofs. The main idea we have followed is to replace all the partial operators with total operators, so that the user no longer has to manipulate dependent types. Once all the theorems and especially rewriting rules have non-dependent hypotheses, they become much easier to apply, since reasoning is back to being backward: from the goal to the hypotheses.

The standard libraries of HOL Light and Isabelle/HOL also provide such total operators for derivatives and integrals. The main difference with our work is that they are defined thanks to Hilbert ε operator, which is not available in Coq. Instead, we have provided algorithms for these operators. They are not the kind of algorithms that one would find in traditional constructive mathematics, since our model of computation is a bit unusual. It has a decidable order, as in the Real RAM model, but it also has a supremum operator and an integer part.

To obtain this model, we have not added any axiom, we have just reused the axiomatization of real numbers from the Coq standard library. We have

also taken great care to never use the axiom of excluded middle, contrarily to what is done in the standard formalization. Unfortunately, when looking at the assumptions of some of the theorems of our library, there might be occurrences of this axiom. They leak from the standard library through the equivalence lemmas between our definition of integral and the standard one. Work is under way to remove these uses from the standard library and therefore get a formalization that no longer relies on excluded middle.

Our development of more than 500 lemmas provides total operators for limits, derivatives, and integrals, and equivalence lemmas between our constructive definitions and the partial operators from the standard library. We have also extended the theory of real analysis further than what is available in the standard library: iterated partial derivatives, parametric integrals, and so on.

We have applied our formalization to filling the holes in the formal verification of a numerical program: the three-point scheme for solving the one-dimensional wave equation. The original formalization set as an axiom the existence of total operators for partial derivatives; the verification would never have succeeded if it had had to cope with the dependent types needed for expressing fourth derivatives. The work presented in this paper fills this gap.

The original formalization was also assuming the existence and the regularity of a solution to the partial differential equations. When starting from d'Alembert formula, the formal proof of these properties is mostly mechanical. One just has to differentiate the formula as many times as needed (up to 20 times for order-4 regularity). For that purpose, we have also developed a reflexive Coq tactic that is able to perform such repetitive tasks. The strength of our tactic is that it is able to differentiate under the integral sign. As far as we know, no similar strategy has been developed for other provers.

Future Works

For this work, we choose Riemann's definition of integral. Our only motivation for this choice is that it is the integral provided by Coq's standard library and we wanted to check that we were not less expressive than the standard library. If not for this constraint, we would have chosen a different definition, *e.g.* Lebesgue integral. Indeed, compared to other definitions of integral, Riemann integral does not have much positive points, except for its prestigious name. Lebesgue integral would have been easier for us to define and to manipulate, since it is almost a simple supremum. Moreover, any Riemann-integrable function is Lebesgue-integrable.

In fact, we could presumably go further than Lebesgue integral and define Henstock–Kurzweil integral. Indeed, our work on compactness has shown that our limited framework (no general excluded middle) was still sufficient to manipulate gauge functions and extract finite subcovers. This is the main property needed to prove Cousin's theorem and therefore to define this integral. It has two main positive points. First, it is no more complicated than Riemann integral. Second, the second fundamental theorem of calculus can now be expressed with-

out any precondition: each differentiable function is the integral of its derivative. This makes its usage in formal proofs straightforward.

For now, we have only defined limits, derivatives, and integrals, as total operators. Our goal is to extend this paradigm to other common operators, *e.g.* power series and reciprocal, so as to provide all the basic blocks of analysis. We also intend to extend our automated tools beyond just differentiation. An obvious extension is integration, but also automatic proofs of integrability and continuity. Indeed, differentiating under the integral sign tends to generate numerous side conditions about these properties and they would greatly benefit from being automatically discharged by the prover during the symbolic computations.

References

1. Boldo, S.: Floats & Ropes: a case study for formal numerical program verification. In: 36th International Colloquium on Automata, Languages and Programming. LNCS - ARCoSS, vol. 5556, pp. 91–102. Springer, Rhodos, Greece (Jul 2009)
2. Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P.: Formal proof of a wave equation resolution scheme: the method error. In: Kaufmann, M., Paulson, L.C. (eds.) 1st Interactive Theorem Proving Conference (ITP). LNCS, vol. 6172, pp. 147–162. Springer, Edinburgh, Scotland (2010)
3. Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P.: Wave Equation Numerical Resolution: a Comprehensive Mechanized Proof of a C Program. Journal of Automated Reasoning (2012), Accepted for publication on May 20th, 2012, `http://hal.inria.fr/hal-00649240`
4. Butler, R.W.: Formalization of the integral calculus in the PVS theorem prover. Tech. rep. (2004)
5. Cruz-Filipe, L.: Constructive Real Analysis: a Type-Theoretical Formalization and Applications. Ph.D. thesis, University of Nijmegen (Apr 2004)
6. Cruz-Filipe, L., Geuvers, H., Wiedijk, F.: C-CoRN : the constructive Coq repository at Nijmegen. In: Mathematical Knowledge Management, Third International Conference, MKM 2004. pp. 88–103 (2004)
7. Fleuriot, J.: On the mechanization of real analysis in Isabelle/HOL. In: Theorem Proving in Higher Order Logics, pp. 145–161 (2000)
8. Gamboa, R., Gamboa, R.: Continuity and differentiability in acl2. In: Computer-Aided Reasoning: ACL2 Case Studies, chapter 18. Kluwer Academic Press (2000)
9. Gamboa, R.A., Kaufmann, M.: Non-standard analysis in acl2 (2001)
10. Gonthier, G., Mahboubi, A., Tassi, E.: A Small Scale Reflection Extension for the Coq system. Rapport de recherche RR-6455, INRIA (2008), `http://hal.inria.fr/inria-00258384`
11. Lelay, C., Melquiond, G.: Différentiabilité et intégrabilité en Coq. Application à la formule de d'Alembert. In: 23èmes Journées Francophones des Langages Applicatifs. pp. 119–133. Carnac, France (2012)
12. Spitters, B., van der Weegen, E.: Type classes for mathematics in type theory. CoRR abs/1102.1323 (2011)