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

Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond

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 scheme over a discrete grid and its convergence proof, meaning that the approximation improves when the grid size decreases.

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 differentiable 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. The developed library can be downloaded and browsed at

http://coquelicot.saclay.inria.fr/results.html
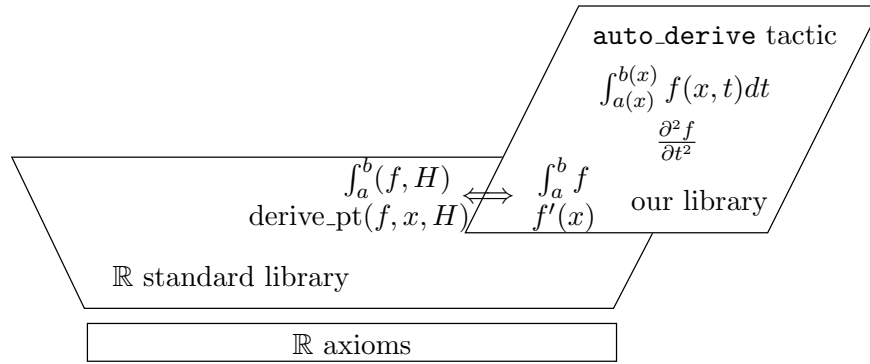
Fig. 1. Sketch of our library: equivalent definitions, additional lemmas, and an automatic tactic, without introducing new axioms.

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.

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], the Coq constructive library C-CoRN[2], HOL Light[3], Isabelle/HOL[4], PVS[5], Mizar[6], and ACL2(r)[7]. We will first present the definitions of the differentiability and integrability predicates, and then describe the design choices for the derivative and integral functions.

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 based on the Landau 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 HOL Light (and Isabelle/HOL which has inherited that formalization of analysis), the Newton's difference quotient is also used, but the limit is a more generic notion as it is defined in a topological space based on a field [12]. Another approach is implemented in additional libraries: the Frechet derivative in a real normed vector space [13]. In C-CoRN, a constructive formalization of real numbers for Coq, the previous definition is modified to get a uniform differentiability [5] on a closed interval $[a;b]$, *i.e.* there is a single δ for all $x \in [a;b]$. In Mizar, differentiability is defined for multi-variable functions [17] as usual by the existence of a linear function L such that $f(x+h) - f(x) - L(h) = o(h)$.

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

$$\forall \varepsilon > 0, \text{ there are two step functions } \varphi, \psi : [a;b] \rightarrow \mathbb{R}, \text{ such that}$$
$$(\forall t \in [a;b], \ |f(t) - \varphi(t)| \leq \psi(t)) \wedge |\textstyle\int \psi| < \varepsilon$$

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

PVS in [4] and Mizar in [7] define Riemann integrability as the convergence of Riemann sums. Both definitions are mathematically equivalent. A difference is that the integral value is explicitly given by this second definition while Coq's definition only provides approximations of Riemann integral.

C-CoRN uses a third equivalent definition of Riemann integrability: the convergence of Darboux sequences. As with Riemann sums, the integral value is directly obtained from the definition.

---

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

[2] `http://corn.cs.ru.nl/`

[3] `http://www.cl.cam.ac.uk/~jrh13/hol-light/`

[4] `http://isabelle.in.tum.de/dist/library/HOL/index.html`

[5] `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/`

[6] `http://www.mizar.org/`

[7] `http://www.cs.utexas.edu/users/moore/acl2/v5-0/`

HOL Light does not define Riemann integrals, but both gauge and Lebesgue integrals, which are more general but less intuitive. As with derivatives, all the notions are defined for multivariate functions.

In a former library about reals in Isabelle/HOL [8], and the current library in ACL2(r) [9], differentiability was defined from non-standard analysis where the formal notion of "infinitely close", *i.e.* difference is less than all positive real numbers, replaces the usual notion of "arbitrarily 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 assistants 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 ACL2(r) [10].

HOL Light 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 [18] is to provide assistance work with these proof terms by trying to infer them.

3 Logical Foundations

Rather than introducing our own set of axioms, we have based our formalization on the axioms from the Coq standard library on real numbers. This ensures that our automated proof tools are usable by works based on the standard library, that some results from the standard library could be reused, and that our work is consistent (assuming the standard library is). We have also taken great care to not introduce any other axioms, especially not the excluded middle, which is neither a native concept in Coq logic, nor a consequence of Coq axioms on real numbers.

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 *non-informative*, that is, one can only use the witness of an existential property $\exists x,\ P(x)$ inside the proof of logical formulas. 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 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 `derivable_pt`$(f, x) \wedge$ `derivable_pt`(g, x) will be rejected by Coq. Indeed, this formula is ill-typed since neither members are logical formulas. A coercion from specification types to formulas in `Prop` would avoid this issue as long as one only needs the differentiability property and not the derivative of a function.

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 end this overview, one should mention that subsets of a type T are usually represented by predicates, that is, functions of type $T \to$ `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 definitional. 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. Except perhaps for the choice for the domain of $\square^{-1}$, 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 `up : R → Z` such that $\forall x \in$ R, $x < $ `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 : $ R $\mid y$ is an upper bound of E 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[8] has shown that it was often unneeded. So we restrict our use of classical reasoning to the proof of goals that are the negation of a logical formula.

Notice also that the `completeness` axiom has an intuitionistic rather than classical feel to it. Indeed, the axiom is written in such a way that it cannot produce a proof of $E(x)$ for some real x (especially not the real that is the supremum of E), since properties of the form $E(x)$ always appear as premises in the axiom. At best, one can derive the following property which protects the existential quantifier behind a double negation:

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

3.3 Limited Principle of Omniscience

Let P be a decidable predicate on natural numbers. The limited principle of omniscience (LPO) 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 the LPO is as follows. Since P is decidable, we can build a function $f(n)$ that returns $1/(n+1)$ if $P(n)$ holds and 0 otherwise. Let us consider the subset of real numbers $\{f(n) \mid n \in \mathbb{N}\}$. 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 inverse (`archimed`) which will act as a witness for building a value of type $\{n \mid P(n)\}$.

---

[8] http://coqtail.sourceforge.net/

This proof of the LPO, inspired by the CoqTail project, is an improvement over some previous work that depended on the `not_all_ex_not` consequence of the excluded-middle axiom [15].

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 the LPO to decide the following alternative (Lemma `Rbar_ub_dec` of our development):

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

The proof requires that $\forall n, \ s_n \leq M$ is decidable, which is just a consequence of the LPO applied to the decidable predicate $n \mapsto M < s_n$.

Let us define the complete lattice $\texttt{Rbar} = \texttt{R} \cup \{-\infty, +\infty\}$, inspired by [14]. Since we are able to decide whether a set is bounded, we can define supremum and infimum functions for nonempty subset of `Rbar`.[9] Let us consider a sequence $(u_n)_{n \in \mathbb{N}}$ of elements of `R` (or `Rbar`). The set of its values is nonempty, thus we can define its superior and inferior limits:

$$\limsup (u_n)_{n \in \mathbb{N}} = \inf_{m \in \mathbb{N}} (\sup_{n \in \mathbb{N}} u_{m+n}) \qquad \liminf (u_n)_{n \in \mathbb{N}} = \sup_{m \in \mathbb{N}} (\inf_{n \in \mathbb{N}} u_{m+n})$$

At this point, we arbitrarily define a function `Lim_seq` from sequences to real numbers as $(\limsup(u_n) + \liminf(u_n))/2$, or 0 in case of infinities. If $(u_n)_n$ is a converging sequence, $\texttt{Lim\_seq}(u_n)_{n \in \mathbb{N}}$ 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. With this definition, we also get "limits value" for non-convergent sequences: $\texttt{Lim\_seq}(-1)^n = (1 + (-1))/2 = 0$.

There were simpler possibilities for defining `Lim_seq`, but this one offers the equality $\texttt{Lim\_seq}(\alpha \cdot u_n) = \alpha \cdot \texttt{Lim\_seq}(u_n)$ for any real α without requiring $(u_n)_n$ to converge.

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

$$\texttt{Lim}(f, x) = \texttt{Lim\_seq} \left(f \left(x + \frac{1}{n} \right) \right)_{n \in \mathbb{N}}.$$

---

[9] Supremum and infimum of a subset P are extended with $\sup \emptyset = -\infty$ and $\inf \emptyset = +\infty$, when one can prove $P \neq \emptyset \Rightarrow \exists x, P(x)$. This generalization is not needed here.

Again, if the function has a limit at this point, in sense of the usual $\varepsilon - \delta$ definition for pointed limit, operator `Lim` returns it, since $\lim_{u \to x} f(u)$ is then equal to $\lim_{n \to \infty} f(u_n)$ for any sequence such that $\lim(u_n) = x$ and $\forall n \in \mathbb{N}$, $u_n \neq x$. Otherwise it does return a value, but a meaningless one. We did not gain anything significant by defining the limit with more complicated sequences such as $(f(x - 1/n) + f(x + 1/n))/2$.

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 set 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 to 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

$$\delta' = \frac{1}{2} \sup \ \{d \mid d \leq 1 \wedge \forall x \in [a, b], \ \exists t \in [a, b], \ |x - t| < \delta(t) \wedge d \leq \delta(t)\}.$$

First, we prove that this supremum is not equal to zero. This is the same idea as with most cover-based proofs: consider the widest interval $[a, b')$ such that the property $\exists d > 0, \ \forall x \in [a, b'), \ \exists t \in [a, b], \ |x - t| < \delta(t) \wedge d \leq \delta(t)$ holds. The bound b' is defined as a supremum value and we prove that, if $b' < b$, there is a contradiction. Then we prove that it still holds for the whole interval $[a, b]$, so δ' is positive.

Second, we prove that δ' satisfies the original property. This is immediate, but only because the property contains a double negation. Indeed, as mentioned before, the completeness axiom never provides us with proofs that the property holds for values smaller than the supremum. So the best we can prove is that there would be contraction if δ' did not satisfy the property. In practice, this double negation does not matter because we always use the compactness property to exhibit contradictions.

Finally, we have proved the compactness property not just for segments of the real line but for any n-orthotope (Cartesian product of n segments).

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:

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

and we proved that, if the derivative number $f'(x)$ exists, it is equal to our derivative number using lemmas proved in the standard library.

This definition allows to write properties about derivative without any proof term. For example to write "the function f is differentiable on the domain D and f' satisfies the property P on this domain" in Coq, we previously had to write

```
Lemma pr : forall x, D x → derivable_pt f x.
Goal forall x (Dx : D x), P (derive_pt f x (pr x Dx)).
```

using a dependent pair. With our approach, we can write the same statement as

```
Goal forall x, D x → ex_derive f x.
Goal forall x, D x → P (Derive f x).
```

and then we can prove separately differentiability and the property P. Moreover, as our derivative is a limit, some properties such as $(\alpha \cdot f)' = \alpha \cdot f'$ do not require the differentiability of f. Therefore, the proof burden is lightened, as theorems have less preconditions. Such a property could also be obtained for a derivative defined on top of Hilbert's epsilon, but not without introducing a cumbersome definition based on conditionals. Our approach also makes it possible to express more easily the n-th derivative:

$$\begin{cases} \texttt{Derive\_n}\,(f, 0, x) \quad\;\; = f(x) \\ \texttt{Derive\_n}\,(f, n+1, x) = \texttt{Derive}\,(x \mapsto \texttt{Derive\_n}\,(f, n, x), x) \end{cases}$$

and we express the n-th differentiability as

```
forall i : nat, (i < n)%nat → ex_Derive (Derive_n f i) x
```

4.2 Riemann integral and Riemann sums

We define our integral `RInt` as the limit of

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

where $x_k^n = a + k \cdot (b-a)/(n+1)$. As above, this definition allows to prove $\forall \alpha \in \mathbb{R}$, $\texttt{RInt}\,(\alpha \cdot f, a, b) = \alpha \cdot \texttt{RInt}\,(f, a, b)$ without hypotheses on f.

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

We chose to define new step functions based on Ssreflect sequences [11]:

```
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. We define our step functions with a generic type `T` to use it in the same way with `T := R` or `T := Rbar`. For example, to define the step function needed for `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 and `SF_seq_f2` builds the needed step function.

Moreover, as the standard library does not provide a global relation between Riemann integrability and Riemann integral, we chose to prove the equivalence between this definition and the following convergence of Riemann sums:

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

where $S(f, \sigma, \xi) = \sum_{k=0}^{n} f(\xi_k) \cdot (\sigma_{k+1} - \sigma_k)$ is a Riemann sum, n is the length of ξ, $n + 1$ the length of σ, $\sigma_0 = a$, $\sigma_{n+1} = b$ and (σ, ξ) is a pointed subdivision, i.e. $\forall k \le n,\ \sigma_k \le \xi_k \le \sigma_{k+1}$. This is the same approach as in PVS and Mizar.

Thanks to this new definition, we can express integrability with the same structure as convergence and differentiability: a function gives the value of the Riemann integral, a predicate states the property of Riemann integrability, and `is_RInt`(f, a, b, If) states that If is the Riemann integral of f between a and b.

We can note that our value is the limit of a sequence of specific Riemann sums $S(f, \sigma_n, \xi_n)$ such that $\forall i \in [\![0; n]\!],\ \sigma_{i+1} - \sigma_i = (b - a)/(n + 1)$. The correctness is then immediate for our new definition of Riemann integrability.

To ensure the compatibility with the standard library of Coq, we proved the equivalence between both definitions, so that we can take advantage of existing lemmas.

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

$$\forall t \geq 0, \ \forall x \in [x_{\min}, x_{\max}], \ \frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = f(x,t),$$

$$\forall x \in [x_{\min}, x_{\max}], \ \frac{\partial u}{\partial t}(x,0) = u_1(x),$$

$$\forall x \in [x_{\min}, x_{\max}], \ u(x,0) = u_0(x),$$

$$\forall t \geq 0, \ u(x_{\min}, t) = u(x_{\max}, t) = 0$$

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 $\mathrm{diff}(z \to \mathrm{diff}(y \to u(x,y), z, H_1), t, H_2)$ 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 for a finite rope [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 $\forall \varepsilon, \exists \delta \dots$ formula). This fully specifies the derivatives but was impractical and difficult to read. Yet, this meant an equivalence 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 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 [16]. 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. As the former one, this tactic is programmed in `Ltac`.

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 library. 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 chose Riemann's definition of integral. Our only motivation for this choice was 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.

In fact, we could presumably go further than Lebesgue integral and define gauge 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 without 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. Rhodos, Greece (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. 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. Journal of Formalized Reasoning 2(1), 1–26 (2009)
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: 3th International Conference on Mathematical Knowledge Management (MKM). pp. 88–103. Bialowieza, Poland (2004)
7. Endou, N., Korniłowicz, A.: The definition of the Riemann definite integral and some related lemmas. Journal of Formalized Mathematics 8(1), 93–102 (1999)
8. Fleuriot, J.: On the mechanization of real analysis in Isabelle/HOL. In: Mark Aagaard, J.H. (ed.) 13th International Conference on Theorem Proving in Higher Order Logics (TPHOL). LNCS, vol. 1869, pp. 145–161. Portland, OR, USA (2000)
9. Gamboa, R.: Continuity and differentiability in ACL2. In: Computer-Aided Reasoning: ACL2 Case Studies, chap. 18. Kluwer Academic Publisher (2000)
10. Gamboa, R., Kaufmann, M.: Non-standard analysis in ACL2. Journal of Automated Reasoning 27(4), 323–351 (2001)
11. Gonthier, G., Mahboubi, A., Tassi, E.: A small scale reflection extension for the Coq system. Tech. Rep. RR-6455, INRIA (2008), `http://hal.inria.fr/inria-00258384`
12. Harrison, J.: Theorem Proving with the Real Numbers. Springer-Verlag (1998)
13. Harrison, J.: A HOL theory of Euclidean space. In: 18th International Conference on Theorem Proving in Higher Order Logics (TPHOL). pp. 114–129. Oxford, UK (2005)
14. Hölzl, J., Heller, A.: Three chapters of measure theory in Isabelle/HOL. In: 2nd Interactive Theorem Proving Conference (ITP). LNCS, vol. 6898, pp. 135–151. Nijmegen, The Netherlands (2011)
15. Kaliszyk, C., O'Connor, R.: Computing with classical real numbers. Journal of Formalized Reasoning 2(1), 27–39 (2009)
16. 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)
17. Raczkowski, K., Sadowski, P.: Real function differentiability. Journal of Formalized Mathematics 1(4), 797–801 (1990)
18. Spitters, B., van der Weegen, E.: Type classes for mathematics in type theory. Mathematical Structures in Computer Sciences 21(4), 795–825 (2011)