Coquelicot: A User-Friendly Library of Real Analysis for Coq

Sylvie Boldo, Catherine Lelay and Guillaume Melquiond

Abstract. Real analysis is pervasive to many applications, if only because it is a suitable tool for modeling physical or socio-economical systems. As such, its support is warranted in proof assistants, so that the users have a way to formally verify mathematical theorems and correctness of critical systems. The Coq system comes with an axiomatization of standard real numbers and a library of theorems on real analysis. Unfortunately, this standard library is lacking some widely used results. For instance, power series are not developed further than their definition. Moreover, the definitions of integrals and derivatives are based on dependent types, which make them especially cumbersome to use in practice. To palliate these inadequacies, we have designed a user-friendly library: Coquelicot. An easier way of writing formulas and theorem statements is achieved by relying on total functions in place of dependent types for limits, derivatives, integrals, power series, and so on. To help with the proof process, the library comes with a comprehensive set of theorems that cover not only these notions, but also some extensions such as parametric integrals, two-dimensional differentiability, asymptotic behaviors. It also offers some automations for performing differentiability proofs. Moreover, Coquelicot is a conservative extension of Coq's standard library and we provide correspondence theorems between the two libraries. We have exercised the library on several use cases: in an exam at university entry level, for the definitions and properties of Bessel functions, and for the solution of the one-dimensional wave equation.

Keywords. Standard real analysis, Coq proof assistant, Library, Generalized limits, Differentiability, Parametric integrals, Power series.

1. Introduction

From physics to economy toward chemistry and combinatorial, many phenomena may be modeled by ordinary or partial differential equations. Unfortunately, studying and solving such equations is not a trivial matter and any error may have dramatic consequences. Computer algebra systems and other related tools may be helpful for uncovering such errors. For higher confidence, one can write detailed proofs of correctness and check them using formal systems. In fact, many proof assistants, such as Mizar, HOL, PVS, provide a formalization of real analysis. These tools help to find errors, but their use is often tedious for non-specialists.

The main difficulty when using these tools in the setting of real analysis stems from the gap between the traditional pen-and-paper proofs and the formal ones. For instance, theorem statements may be less readable once formalized. Moreover, lemmas from libraries of real analysis are often inadequate, *e.g.* because of contrived hypotheses. This makes them unpractical to apply and leads to proofs longer than they should be.

This work was supported by Project Coquelicot from RTRA Digiteo and Région Île-de-France.

Our system of interest is the Coq proof assistant and we would like to use it for proving results of real analysis, *e.g.* about partial differential equations [5]. This calls for a user-friendly library that provides limits of sequences and functions, derivatives, integrals, power series, and numerous theorems that relate these notions. It should be noted that Coq already provides a library of real analysis in its distribution. This formalization started with Micaela Mayero's PhD in 2001 [23]. It was later extended by Olivier Desmettre. It is based on a classical axiomatization of real numbers and provides the usual definitions of standard analysis such as finite limits of sequences and univariate functions, derivatives, Riemann integrals, and power series.

This library suffers from several shortcomings though. First, it has not evolved much since its inception, so it does not support some modern features of Coq, such as type classes. Another issue is the lack of homogeneity: some definitions and theorems are missing, naming policy is chaotic, useless hypotheses are creeping in theorem statements. For instance, arithmetic operations on power series are not provided; theorems about Riemann integrals are inconveniently named `RiemannInt_P1` to `_P33`; monotonicity of square root $\sqrt{u} < \sqrt{v}$ requires the redundant hypotheses $0 \leq u$, $0 \leq v$, and $u < v$.

The previous issues make the library a bit unfriendly, but they can be overcome with some practice. A more salient issue is the overall use of dependent types for defining some notions. For instance, differentiability is provided by the predicate `derivable_pt : (R -> R) -> R -> Set` while the differentiation operator has the following type:

```
derive_pt : ∀ (f : R → R) (x : R), derivable_pt f x → R.
```

In other words, this operator takes a function f, a real number x, and a proof that f is differentiable at point x, and it returns the value of the derivative of f at x. This has several consequences. First, any statement about the derivative of a function has to embed a previously proved lemma about the differentiability of that function. Second, rewriting an expression such as $(f + g)'(x)$ into $f'(x) + g'(x)$, while straightforward in mathematics, is next to impossible to achieve in Coq, as the proof terms have to be manually built by the user beforehand. Third, these proof terms can quickly blow up, and thus become impossible for the user to exhibit, *e.g.* in the case of iterated derivatives or differentiation under an integral. By the way, integrals in Coq suffer from the same issues. Notice also that the predicates for differentiability and integrability are not in `Prop`, which is the sort of logical propositions in Coq. This peculiarity makes them especially unwieldy in practice.

The standard library is not the sole attempt at formalizing real analysis in Coq. Another mature project is the C-CoRN / MathClasses library [14], which provides Bishop-like constructive mathematics. Coq is especially well suited for such a development, as its logic itself is constructive, contrarily to most other proof assistants. C-CoRN provides even more results than Coq's standard library. It is not a suitable foundation for our project of building a real analysis library though. Indeed, constructive analysis is far from the standard classical one. For instance, every function is continuous, which might preclude some applications.

Another way for building a real analysis library for Coq would be to port John Harrison's formalization [17]. It encompasses a large piece of real analysis and is now pervasive in most HOL-like systems, *e.g.* HOL Light and Isabelle/HOL. Unfortunately, it is built upon axioms specific to HOL systems, such as excluded middle, and we preferred not to add them, however compatible they are with Coq's logic. So we could not use it as a foundation either, even if it was inspirational.

Therefore, neither of the two real analysis libraries already available in Coq are suitable for our purpose, and porting a library from another system is not an option either. These considerations led to the Coquelicot project. Our goal was to design a user-friendly formalization of real analysis in Coq. There were two additional constraints. First, it had to be a conservative extension of the standard library: no new axioms had to be introduced. Second, it had to use the existing definitions or provide equivalent ones, so that all the tools built on top of the standard library could be reused transparently (*e.g.* tactics such as `field` or `lra`).

This paper presents version 1.1 of the Coquelicot library. It is available at

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

The main features of the library are as follows. It defines total functions for expressing derivatives and integrals without dependent types. It generalizes limits to the set $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. It provides theorems for doing analysis on multivariate functions, *e.g.* parametric integrals. It also provides some automation for performing proofs of differentiability.

Section 2 gives an overview of the real analysis libraries available for some of the predominant formal systems. Section 3 describes the basic building blocks of our library: axioms, limited principle of omniscience, compactness, and local properties. Section 4 presents the total functions, predicates, and lemmas, available for the basic notions of real analysis: limits, derivatives, integrals, power series. Section 5 shows some of the advanced features of the library: partial derivatives, parametric integrals, generalized limits, and automation. Section 6 illustrates the use of Coquelicot on a few applications: in an exam at university entry level, for the definitions and properties of Bessel functions, and for the solution of the one-dimensional wave equation.

2. State of the Art

As alluded, there are numerous proofs assistants and many of them have a formalization for real numbers and standard analysis. The proof assistants and their libraries will be described to give an idea of their respective power and specificities. For more details, we refer the reader to [7].

We choose to skim over non-standard analysis as it is out of the scope of this paper. Non-standard analysis is implemented in an extension of the first-order ACL2 system called ACL2(r) which offers support for reasoning about irrational and complex numbers. It modifies the ACL2 logic by introducing notions from non-standard analysis and thus avoids quantifiers [13]. There are several shortcomings that prevent any advanced usage of the real analysis library of ACL2(r). Indeed, the absence of higher-order logic makes it difficult to manipulate functions. It is even more difficult to define partial functions. Finally, there is no definition of any integral. The Isabelle/HOL library also provides non-standard analysis in addition to the standard one. Contrarily to the axiomatic approach of ACL2(r), hyper-reals are defined as equivalence classes of sequences of real numbers [12].

For the sake of significance, we restrict ourselves to systems that stood the test of time, meaning they are decades old. Moreover they have to come with a standard library for real arithmetic, and to provide features for real analysis (and not just a construction of real numbers). The standard libraries we have picked out are those of Mizar, PVS, HOL Light, and Isabelle/HOL. For Coq, we consider both the standard library and the C-CoRN/MathClasses library.

2.1. Mizar

Mizar[1] [33, 27] has for goal that the proofs be close to the mathematical vernacular, so that mathematicians may easily use it. This makes the proofs longer but somehow more readable. The Mizar Mathematical Library is huge: 9,400 definitions of mathematical concepts and more than 49,000 theorems, that have been accumulated since 1989. In particular, it contains definitions that seldom appear in other provers, such as limit at infinity, one-side limits, Riemann integral, differentiability for multivariate functions. The drawback is that Mizar is a fixed system: it cannot be extended or programmed by the user. There is no computational power nor user automation.

While originally an axiomatization, the formalization of real numbers in Mizar was later changed to a construction based on Dedekind cuts [34]. Since Mizar is based on set theory, Dedekind cuts easily fit in this system and having functions defined on partial domains is straightforward. Unfortunately, this set-theoretical approach is not always the most natural to use, and this makes the statements and proofs less user-friendly.

[1] `http://mizar.org/`

2.2. PVS

PVS[2] is a formal system based on classical higher-order logic [28]. It heavily uses predicate subtypes and dependent types [32]; the TCCs (type-correctness conditions) are proof obligations generated by the PVS typechecker, for example to prevent division by zero. Usability and automations are polished with efficient decision procedures and a large use of typechecking information for the proofs. A large PVS library is maintained by the NASA, and this is where one has to look for real analysis results.

The PVS system postulates real numbers as a complete Archimedean field. The axioms for number fields are straightforward. Nine of them give the properties of addition, multiplication, and their inverses. The last two define subtraction and division from opposite and inverse. Note that no axiom states $0 \neq 1$. Indeed, this property comes for free, since PVS integers are mapped to Lisp integers. In addition to these axioms defining an ordered field, PVS theories postulate the completeness of $\mathbb{R}$ by the existence of the least upper bound for any nonempty upper-bounded subset of real numbers [11].

Many real analysis results are present (even two definitions of the real exponential function), notably including power series, both Riemann and Lebesgue integrals. Partial functions are easy to define using sub-typing. Many automations are available including the Manip package for an easy handling of equalities and inequalities, and a `FIELD` strategy for equalities on reals. The basic and efficient `GRIND` strategy can be extended with additional theories for reals. PVS also provides some strategies based on numerical computations: `numerical` performs interval arithmetic to verify inequalities involving transcendental functions [10]; `bernstein` performs global optimization based on Bernstein polynomials to verify systems of polynomial inequalities [26].

2.3. HOL Light and Isabelle/HOL

We now present the real analysis found in various provers that inherit the original HOL prover: mainly HOL Light[3] and Isabelle/HOL[4] , but also HOL4 and ProofPower-HOL. Most of them are based on classical higher-order logic with axioms of infinity, extensionality, and choice in the form of Hilbert's operator.

Even if the definition of real numbers may be different, the real analysis library are all inspired by HOL Light's one, developed by John Harrison [16]. Here, functions are total and theorems work only on restricted domains. The Isabelle/HOL library shows the benefits of using filters instead of nets for expressing limits [18]; it also benefits from type classes.

In HOL Light, real numbers are defined using nearly-additive sequences of natural numbers [15]. The existence of the least upper bound of any nonempty bounded set of nearly-additive sequences is then proved; this theorem is later extended to give the completeness of real numbers. HOL4 and ProofPower-HOL define real numbers using Dedekind cuts. In Isabelle/HOL, the formalization relies on Cauchy sequences of rational numbers, even though the original version relied on Dedekind cuts.

An advanced point in those provers is the intensive use of topological spaces for definitions, such as the convergence of sequences and functions, differentiability and uniform convergence. Power series are defined and many results are proved (including differentiability in HOL4 only). All these provers provide a definition of the Henstock-Kurzweil integral, or gauge integral, while Isabelle/HOL also provides Lebesgue integral.

A lot of automations are available in those systems. Equalities on reals are handled by `REAL_RING` and `REAL_FIELD` in HOL Light; polynomial equalities by `algebra` in Isabelle/HOL (using Gröbner basis). Some tactics provide quantifier elimination: `REAL_ARITH` in HOL Light uses Fourier-Motzkin quantifier elimination for solving universally-quantified systems of linear inequalities; `REAL_ELIM_CONV` in HOL Light is a quantifier-elimination procedure for real arithmetic on

[2] `http://pvs.csl.sri.com/`
[3] `http://www.cl.cam.ac.uk/users/jrh/hol-light/index.html`
[4] `http://isabelle.in.tum.de/`

multivariate polynomials [24]; `ferrack` in Isabelle/HOL provides a quantifier elimination inspired by Ferrante and Rackoff's algorithm. A decision procedure based on *Positivstellensatz* refutations and sums of squares is also available in HOL Light and Isabelle/HOL.

2.4. Coq Standard Library

As our library, the next two libraries use the Coq[5] formal language. Coq is based on the Calculus of Inductive Constructions which combines both a higher-order logic and a richly-typed functional programming language [3]. This logic lacks several axioms that are available to many other formal systems. These include extensionality, Hilbert's operator ε or its variant the ι operator, and excluded middle. Moreover, even if a predicate satisfies the excluded-middle property, Coq does not allow its truth value to be tested inside the body of a function; it can be decided only inside proofs.

The Coq library is structured into two parts: the initial library, which contains elementary logical notions and data-types, and the standard library, a general-purpose library containing various developments and axiomatizations about sets, lists, sorting, arithmetic, real numbers, etc. Real numbers from the standard library are axiomatic; their axioms are detailed in Section 3.1. Here we describe only a few notions of the real analysis library.

Riemann integrability is defined with step functions and the traditional ε-δ definition:

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

The value of the integral is then defined as the limit of $\int \varphi$ when $\varepsilon \to 0$. The standard library also provides the definition of Newton's integral, that is, a function g is Newton integrable if there is a function f differentiable such that $f' = g$. As mentioned, Riemann integrals (resp. derivatives) have dependent types, so expressing them requires to exhibit terms proving that the functions are integrable (resp. differentiable).

The Coq proof assistant comes with numerous tactics. First `ring` makes it possible to automatically prove equalities between two polynomials, and the `field` variant of the tactic can cope with divisions. For proving polynomial equalities using the context, there is `nsatz` [31] and `psatz` that explores cones by increasing degrees [4]. Coq also provides the `fourier` tactic (superseded by `lra` in recent versions) based on Fourier-Motzkin quantifier elimination for solving universally-quantified systems of linear inequalities.

2.5. Another Coq Library: C-Corn/MathClasses

Proof developments at Nijmegen led to a different library called C-CoRN [9] and its successor MathClasses [20]. The main idea is to provide Bishop-like constructive mathematics. These libraries are quite large and generic. For example, the convergence of a real sequence works in any ordered field and not just real numbers.

The C-CoRN library for Coq constructs real numbers on top of Cauchy sequences [14]. But Coq, contrarily to HOL-style provers, has poor support for quotient types, so the whole C-CoRN formalization is built on the notion of *setoid*. Another distinguishing feature is that the real analysis of C-CoRN is not built directly upon the setoid of real numbers built from Cauchy sequences, but rather on an abstract type that satisfies a given signature `CReals`.

The MathClasses library is based on the completion monad $\mathfrak{C}$ defined in [29]. It is a monad on the category of metric spaces with uniformly continuous functions. $\mathbb{R}$ is then defined as $\mathfrak{C}(\mathbb{Q})$. A real number x is defined as being nonnegative when

$$\forall \varepsilon : \mathbb{Q}^+, \quad -\varepsilon \leq_{\mathbb{Q}} x(\varepsilon),$$

where $x(\varepsilon)$ is a rational number that approximates x within the distance ε. The order $x \leq_{\mathbb{R}} y$ is then defined as $y - x$ nonnegative. Abstract interfaces are heavily used to ease statements and

[5] `http://coq.inria.fr/`

proofs [20]. Thanks to type classes, the algebraic and order hierarchies (setoid, group, ring, and so on) easily benefit from inheritance.

In C-CoRN, functions have to be compatible with the equivalence relation, so that a function applied to different Cauchy sequences representing the same real number gives equivalent results. So each function has to come with this property of being well-defined, which forces it to be naturally continuous. C-CoRN defines Riemann integral for uniformly continuous functions as the limit of the sequence of Riemann sums. The library provides no notion of Riemann integrability because all functions are continuous thus Riemann integrable. MathClasses also defines Riemann integral, and even Stieltjes integral [30]. First, step functions and partitions are defined. Then, using several monads including the completion monad, step functions are lifted to define integrable functions. As with other constructs of MathClasses, the integral can be effectively computed.

The computational power of these libraries allows to prove polynomial and transcendental inequalities just by computing the values with enough precision as long as there are no indeterminates.

3. Basic Blocks of the Library

Our library is a conservative extension of Coq's standard library on real numbers. In this section, we detail what the underlying axioms are. We also show how we built the two main tools for doing analysis: the ability to compute limits and the ability to extract finite coverings from compact sets. Finally, we present our take on ε-δ reasoning.

3.1. 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 assumed 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 $\square^{-1}$, these axioms are not controversial.

To get the proper real line, one also needs this field to be Archimedean and closed under the supremum bound. A peculiarity of Coq's standard library comes from the fact that these axioms are expressed by two functions: one can compute the integer part of a real number, the other can compute the supremum of an upper-bounded subset of `R`. Moreover, there is a third function, which is able to compare two real numbers. Below are the actual definitions from the library:

- Lemma `archimed` states that $\texttt{up} : \texttt{R} \to \texttt{Z}$ satisfies $\forall x \in \texttt{R},\ x < \texttt{up}(x) \leq x + 1$.
- Given a subset E of `R` and some proofs that it is both inhabited ($\exists x,\ x \in E$) and bounded ($\exists M,\ \forall x,\ x \in E \Rightarrow x \leq M$), function `completeness` returns a real that is the least upper bound of E. (It does not, however, provide a real x such that $x \in E$.)
- Given two real numbers x and y, function `total_order_T` tells whether $x < y$ or $x = y$ or $x > y$. This is equivalent to the decidability of the equality on real numbers.

Note that the standard library sometimes makes use of the excluded-middle axiom in addition to the previous ones. The CoqTail project[6], however, has shown that it was unneeded. So, for the purpose of this work, we consider that this axiom is not part of the ones defining real numbers in Coq's standard library. We will not use it, nor we will use any other axioms that we could have defined ourselves or found in some dark corner of the standard library.

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

3.2. Limited Principle of Omniscience

The conjunction of `completeness` and `total_order_T` causes any formula that satisfies the excluded-middle principle to become decidable. This strong property is of little interest for our development though. For doing real analysis, one can derive a more useful property from the axioms defining real numbers in Coq: the limited principle of omniscience (LPO).

Let P be a decidable predicate on natural numbers. The LPO states that one can decide whether the property never holds. Moreover, if $P(n)$ happens to hold for some number n, the LPO produces such a number. The original idea of the proof comes from [19]; the CoqTail project later improved it by removing the need for the `not_all_ex_not` consequence of the excluded-middle axiom. We have improved it further by getting rid of the sizable amount of analysis it needed (geometric series, logarithm, and so on).

Let us sketch our Coq proof. 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 its supremum is given by `completeness`. This supremum can be tested against 0 by `total_order_T`. If it is zero, we deduce $\forall n,\ \neg P(n)$. Otherwise we compute its discrete inverse with `archimed`, which is a value n such that $P(n)$ holds.

The LPO is not immediately useful for analysis. But by applying it twice in a row, one can deduce whether a sequence of real numbers is bounded, and thus what its supremum and infimum limits are. This paves the way to deciding whether that sequence has a limit and computing it. As limits are pervasive in real analysis, this is one of the basic blocks of our work. More details on those proofs can be found in [6].

3.3. Compactness

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. Our goal is to stay as close as possible to traditional analysis, so dropping compactness is not under consideration.

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 full strength 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 replace the traditional definition by a version related to Cousin covers and gauge functions. Given an interval box $[\mathbf{a}, \mathbf{b}] = [a_1, b_1] \times \cdots \times [a_n, b_n]$ and a gauge function $\delta : \mathbb{R}^n \to \mathbb{R}^+$, function `compactness_value` produces a value $\delta' > 0$ such that

$$\forall x \in [\mathbf{a}, \mathbf{b}],\ \neg\neg\exists t \in [\vec{a}, \vec{b}],\ |x - t| < \delta(t) \wedge \delta' \leq \delta(t).$$

Notice the double negation before $\exists t$. Indeed, while we can compute a value of δ', we have no way of extracting the subcover that actually satisfies it. In practice, this double negation does not hinder us since we always use the compactness property to exhibit contradictions. Again, more details on those proofs can be found in [6].

3.4. Local Properties

When it comes to doing analysis, an important tool is the ability to reason about properties that hold on neighborhoods. For instance, the notion of continuity of f at x requires, for any $\varepsilon > 0$, to find a neighborhood V of x such that any point u of V satisfies $|f(u) - f(x)| < \varepsilon$. The situation is similar for convergence and differentiability, except that one uses a pointed neighborhood $V \setminus \{x\}$. Rather than using arbitrary neighborhoods, topology tells us that looking at open balls centered at x

is equivalent. This is what the `locally` predicate encodes:

$$\texttt{locally}(x, P) \Leftrightarrow (\exists \delta > 0,\ \forall u \in \mathbb{R},\ |u - x| < \delta \Rightarrow P(u)).$$

Continuity at x then becomes $\forall \varepsilon > 0,\ \texttt{locally}(x, (u \mapsto |f(u) - f(x)| < \varepsilon))$.

While there are ways to perform ε-δ reasoning in Coq by relying on existential variables and delayed instantiation [8], our approach avoids these manipulations entirely. For instance, given two hypotheses $\texttt{locally}(x, P)$ and $\texttt{locally}(x, Q)$, some theorem gives $\texttt{locally}(x, (u \mapsto P(u) \wedge Q(u)))$. The proof of this theorem takes care of constructing the intersection of both initial neighborhoods. Such theorems exist for most logical constructs.

This approach is not generic enough though. In HOL Light, nets are used to generalize $\varepsilon - \delta$ formulas [17], while in Isabelle/HOL, filters are used [18]. We have chosen the latter approach and have definitions similar to those found in Isabelle/HOL. As a consequence, the continuity above can now be expressed as $\forall P,\ \texttt{locally}(f(x), P) \Rightarrow \texttt{locally}(x, P \circ f)$. Notice that the quantifiers on ε and δ are now completely hidden inside the `locally` expressions. Both expressions $\texttt{locally}(f(x))$ and $\texttt{locally}(x)$ are so-called filters.

More generally, any filter can be used in place of $\texttt{locally}(\cdot)$. By choosing them appropriately, we get definitions for continuity, convergence, and so on. On a set T, a filter has type $(T \to \texttt{Prop}) \to \texttt{Prop}$, so it can be interpreted either as a set of subsets of T (intuitively the set of neighborhoods at a given point of T) or as a set of predicates on T (intuitively the set of properties that hold in some neighborhood of a given point). There are two main benefits to using filters in our library. First, it makes it possible to factor numerous proofs. For instance, the limit of the sum of two sequences $\lim(u_n + v_n) = \lim u_n + \lim v_n$, the limit of the sum of two functions $\lim_{t \to x}(f(x) + g(x)) = \lim_{t \to x} f(x) + \lim_{t \to x} g(x)$, the continuity of the sum of two functions $f + g$, are all instances of the same proof about addition. Second, filters make it especially simple to prove properties of composition (*e.g.* $g \circ f$ is continuous when f and g are), since it is just a matter of performing a *modus ponens*.

As mentioned above, $\texttt{locally}(x)$ is a filter. Its actual definition works with any metric space and not just $\mathbb{R}$ though. Another important family of filters in Coquelicot is `Rbar_locally`; it extends `locally` to $\mathbb{R} \cup \{+\infty, -\infty\}$ by adding neighborhoods for infinities, *e.g.*

$$\texttt{Rbar\_locally}(-\infty, P) \Leftrightarrow (\exists M \in \mathbb{R},\ \forall u \in \mathbb{R},\ u < M \Rightarrow P(u)).$$

Finally, the last important filter is `eventually`; it is used to express the convergence of sequences:

$$\texttt{eventually}(P) \Leftrightarrow (\exists N,\ \forall n \in \mathbb{N},\ N \leq n \Rightarrow P(n)).$$

There are some variants of those filters available for specific situations. In particular, one can restrict a filter to only a subset of space T. For instance, the limit at the left of a point x can be defined using the following filter:

$$\texttt{at\_left}(x, P) \Leftrightarrow \texttt{locally}(x, (u \mapsto u < x \Rightarrow P(u))).$$

Coquelicot defines some type classes for filters and metric spaces, so that Coq can infer them whenever they are needed. In particular, the product of two filters is proved to be a filter, which is critical for defining the continuity of bivariate functions. And the product of two metric spaces is also a metric space, *e.g.* $\mathbb{R}^2$.

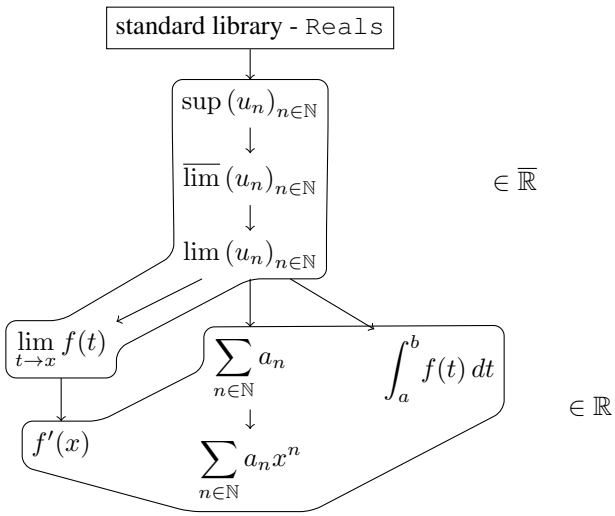

FIGURE 1. Dependencies between total functions.

4. Usual Notions of Real Analysis

In pen-and-paper proofs, it is common to first write functions or formulas and prove their well-formedness afterwards. Unfortunately, the derivative from the standard library is defined using dependent types, so one must first prove the differentiability before being allowed to write the derivative. Our idea is to build total functions for limits, derivatives, and integrals. They return the expected value in the case of convergence, differentiability, or integrability, and else return an arbitrary value.

4.1. Total Functions on Sequences

As illustrated in Figure 1, we first construct total functions for the extrema of sequences $\sup$ and $\inf$, and then for the superior limit $\overline{\lim}$ and the inferior limit $\underline{\lim}$. These four functions have the advantage of being always defined in classical mathematics, without any hypotheses on the input sequence. Thanks to the limited principle of omniscience, described in Section 3.2, we define them in Coq, also without hypotheses. The construction is performed as follows.

We first define a predicate `is_sup_seq` to characterize the extended real number ℓ that is the least upper bound of $u : \mathbb{N} \to \overline{\mathbb{R}}$:

```
match ℓ with
  | ℓ ∈ ℝ   ⇒ ∀ε > 0,  (∀n ∈ ℕ,  u_n <_ℝ̄ ℓ + ε) ∧ (∃n ∈ ℕ,  ℓ − ε <_ℝ̄ u_n)
  |  +∞     ⇒ ∀M ∈ ℝ,  ∃n ∈ ℕ,  M <_ℝ̄ u_n
  |  −∞     ⇒ ∀M ∈ ℝ,  ∀n ∈ ℕ,  u_n <_ℝ̄ M
end.
```

This definition of the least upper bound for the finite case may seem a bit unusual, but it is more practical than the equivalent characterization

$$(\forall n \in \mathbb{N},\ u_n \leq \ell) \wedge (\forall M \in \mathbb{R},\ (\forall n \in \mathbb{N},\ u_n \leq M) \Rightarrow \ell \leq M)$$

because it is closer to the ε-N definition used for superior and inferior limits.

Using the LPO, we have proved that such an extended real number ℓ exists for all sequences of extended real numbers. This lemma is used to build the total function that returns the supremum of a sequence. Function $\inf$ is built similarly.

Superior and inferior limits have also been built using a similar approach: first a predicate to express "$\ell \in \overline{\mathbb{R}}$ is the superior/inferior limit of the sequence $u : \mathbb{N} \to \mathbb{R}$", then lemmas to prove the existence of such ℓ, and finally total functions. We have also proved some lemmas linking these limits with extrema, such as $\texttt{LimSup\_seq}\, u = \inf(n \mapsto \sup(m \mapsto u_{n+m}))$.

4.2. Limit of Sequences and Series

4.2.1. Limits. Limits of sequences are defined for both finite and infinite limits using filters:

```
Definition is_lim_seq (u : nat → R) (l : Rbar) :=
  filterlim u eventually (Rbar_locally l).
```

This definition is useful to inherit properties from filters. It is sometimes less practical than the ε-N one, so we have proved the equivalence with the following formulas:

$$\begin{aligned}
\texttt{is_lim_seq}\,(u,\ell) &\Leftrightarrow \forall \varepsilon > 0,\ \exists N \in \mathbb{N},\ \forall n \in \mathbb{N},\ N \leq n \Rightarrow |u_n - \ell| < \varepsilon \\
\texttt{is_lim_seq}\,(u,+\infty) &\Leftrightarrow \forall M \in \mathbb{R},\ \exists N \in \mathbb{N},\ \forall n \in \mathbb{N},\ N \leq n \Rightarrow M < u_n \\
\texttt{is_lim_seq}\,(u,-\infty) &\Leftrightarrow \forall M \in \mathbb{R},\ \exists N \in \mathbb{N},\ \forall n \in \mathbb{N},\ N \leq n \Rightarrow u_n < M
\end{aligned}$$

where u is a sequence of real numbers and ℓ a real number.

Thanks to our total functions $\overline{\lim}$ and $\underline{\lim}$, the limit of a sequence u can be defined as

$$\texttt{Lim\_seq}\, u := \frac{\overline{\lim}\, u + \underline{\lim}\, u}{2}.$$

We have proved the equivalence between the equality of the superior and inferior limits and the convergence of a sequence. This leads to the correctness of our total function for the limit of convergent sequences:

```
Lemma is_lim_seq_unique : ∀ (u : nat → R) (l : Rbar),
  is_lim_seq u l → Lim_seq u = l.
```

This lemma also gives the uniqueness of the limit by transitivity of equality. In case of non-convergent sequences, $\texttt{Lim\_seq}\, u$ returns an extended real number with no specific properties.

4.2.2. Series of real numbers and power series. Series and power series are good examples of convergence toward a finite number. Indeed, these two kinds of series are defined as the finite limits

$$\texttt{is\_series}\,(a,\ell) \Leftrightarrow \texttt{is\_lim\_seq}\left(\left(n \mapsto \sum_{k=0}^{n} a_k\right), \ell\right)$$

and

$$\texttt{is\_pseries}\,(a,\ell) \Leftrightarrow \texttt{is\_series}\,((n \mapsto a_n x^n), \ell)$$

with a a sequence of real numbers and ℓ a real number. The sum used in these definitions is that of the standard library. Series and power series are then defined as

$$\texttt{Series}\,(a) := \texttt{Lim\_seq}\left(n \mapsto \sum_{k=0}^{n} a_k\right) \text{ and } \texttt{PSeries}\,(a,x) := \texttt{Series}\,(n \mapsto a_n x^n).$$

The convergence radius plays an important role for power series. We define it as follows:

$$\texttt{CV\_radius}\,(a) := \sup\{r \in \mathbb{R} \mid \textstyle\sum |a_n r^n| \text{ is convergent}\}$$

where sup is a total function from nonempty sets to extended real numbers. For better usability, we prove the equality of this number with $\sup\{r \in \mathbb{R} \mid \{|a_n r^n| \ ;\ n \in \mathbb{N}\} \text{ is bounded}\}$.

In contrast with most of the other formalizations, our lemmas about arithmetic and analytic operations on power series do not need any user intervention to rebuild power series and chain lemmas about power series. For example, in most theorem provers, there are no theorems about

multiplication of power series, so the user has to apply those about series and rebuild a power series afterwards, whereas Coquelicot provides

```
Lemma PSeries_mult : ∀ (a b : nat → R) (x : R),
  Rabs x <_R CV_radius a → Rabs x <_R̄ CV_radius b
  → PSeries (PS_mult a b) x = PSeries a x × PSeries b x.
```

where `PS_mult a b` is the Cauchy product of real sequences `a` and `b`.

4.3. Limits of Functions and Derivatives

4.3.1. Limits of functions. As with limits of sequences, limits of functions are defined using filters:

```
Definition is_lim (f : R → R) (x l : Rbar) :=
  filterlim f (Rbar_locally' x) (Rbar_locally l).
```

and this definition is proved equivalent to the usual ε-δ one.

The total function for limits of sequences is used to define the limit of functions without proof terms:

$$\texttt{Lim}\,(f,x) := \texttt{Lim\_seq}\ (f\,([x]_n))_{n\in\mathbb{N}} \in \overline{\mathbb{R}}$$

where $([x]_n)_{n\in\mathbb{N}}$ is the following sequence of real numbers:

$$\forall n \in \mathbb{N},\ \begin{cases} [x]_n & := x + \dfrac{1}{n+1} \\ [+\infty]_n & := n \\ [-\infty]_n & := -n \end{cases}$$

These sequences satisfy $\forall x \in \overline{\mathbb{R}},\ \lim ([x]_n)_{n\in\mathbb{N}} = x$. The correctness of `Lim` is then an application of the composition of a convergent function by a convergent sequence. Again, when f is not convergent at x, the returned value has no specific properties.

Thanks to the `Lim` and `Lim_seq` functions, we can provide an improved theorem about the uniform convergence of function sequences. In particular, contrarily to the version from the standard library, the user does not have to exhibit the limit of the f_n functions:

for all sequence of functions $(f_n)_{n\in\mathbb{N}}$ and all open set $D \subset \mathbb{R}$,
if $(f_n)_{n\in\mathbb{N}}$ is uniformly convergent and $\forall x \in D,\ \forall n \in \mathbb{N},\ \lim\limits_{t\to x} f_n(t)$ exists, then

$$\forall x \in D,\quad \lim_{t\to x}\left(\lim_{n\to\infty} f_n(t)\right) = \lim_{n\to\infty}\left(\lim_{t\to x} f_n(t)\right).$$

4.3.2. Derivative and iterate derivatives. While the previous predicates are defined in Coquelicot and proved equivalent to the ones from the standard library, the definition of differentiability `derivable_pt_lim` directly comes the standard library. For uniformity, we rename it `is_derive` though. The total function for the derivative is defined as the limit of Newton's quotient:

$$\texttt{Derive}\,(f,x) := \texttt{Lim}\ \left(h \mapsto \frac{f(x+h)-f(x)}{h}, 0\right)$$

This function allows to easily define iterate derivatives:

$$\begin{aligned} \texttt{Derive\_n}\ (f,0,x) &:= f(x) \\ \texttt{Derive\_n}\ (f,n+1,x) &:= \texttt{Derive}\ (\texttt{Derive\_n}\,(f,n),x). \end{aligned}$$

Using this function, iterate differentiability $\texttt{is\_derive\_n}(f,n,x,\ell)$ is defined by

```
match n with
| 0   ⇒ f(x) = ℓ
| n+1 ⇒ is_derive (Derive_n (f,n),x,ℓ)
end.
```

Notice that this definition of n-th differentiability does not imply $n-1$-th differentiability. In practice however, this does not matter. For instance, the n-th differentiability between a and b will be stated as

$$\forall x \in \mathbb{R},\ a < x < b,\ \forall k \leq n,\ \texttt{ex\_derive\_n}\ (f, k, x)\,.$$

4.4. Riemann Integral

The total function for Riemann integral is

$$\texttt{RInt}\ (f, a, b) \texttt{ := Lim\_seq} \left(\frac{b-a}{n+1} \sum_{k=0}^{n} f\left(\frac{x_{k+1} + x_k}{2} \right) \right)_{n \in \mathbb{N}}$$

where $x_i = a + \frac{i}{n+1}(b-a)$. As before, we also define two predicates: $\texttt{is\_RInt}\ (f, a, b, If)$ states that the real number If is the integral of f between a and b, while $\texttt{ex\_RInt}\ (f, a, b)$ states the Riemann integrability. We have also proved the equivalence with the standard library definitions.

4.5. Basic Properties

In addition to defining those predicates and functions, we have proved numerous lemmas about them. In particular, we have proved the compatibility of these notions with the basic operations: composition, scalar multiplication, opposite, and addition for function limits, derivatives and integrals; multiplication and multiplicative inverse for limits and derivatives. For limits, these theorems handle limits both at finite and infinite points, and both finite and infinite limit values. (See Section 5.3.)

The way we have defined our total function for limits of sequences allows us to prove rewriting rules that do not require hypotheses:

```
Lemma Lim_seq_scal_l : ∀ (u : nat → R) (a : R),
  Lim_seq (fun n ⇒ a × u n) = a × Lim_seq u.
```

As a consequence, for all the total functions built upon it, opposite and scalar multiplication rewritings can be performed without any constraint. There are some other rewriting rules that do not require any hypothesis: linear composition for Riemann integral $\int_b^a f(ux+v)\,dx = \int_{ub+v}^{ua+v} f(x)\,dx$, and multiplication by a variable for power series $x^k \sum a_n x^n = \sum a_{n-k} x^n$.

Regarding power series, we have also proved that they are differentiable and integrable, and that their convergence radius is left unchanged by these operations. For arithmetic operations, we have proved that the convergence radius of the resulting power series is no smaller.

5. Some Extensions of the Library

Section 4 has shown the low-level definitions provided by the library and the theorems available to the user. The scope of these definitions and theorems roughly covers what the standard library provides. Our library does not stop there though. We also provide some features such as partial derivatives, parametric integrals, limits that uniformly apply to finite and infinite points, asymptotic behaviors of functions, and a few powerful tactics.

5.1. Partial Derivatives

A use case for our library is the study of some systems of partial differential equations. This entails the need for multivariate functions and the ability to manipulate partial derivatives. Given a higher-order language, they can easily be defined from the standard one-dimensional derivative; the partial derivative of f with respect to the k-th argument at point $(x_1, \ldots, x_n)$ is simply $\texttt{Derive}((u \mapsto f(x_1, \ldots, u, \ldots, x_n)), x_k)$. As such, the library does not even provide a definition. It does provide various theorems about partial derivatives though.

Note that the notion of partial derivative is weaker than Frechet's derivative (which can be found in HOL Light and Isabelle/HOL). Indeed, the existence of Frechet's derivative implies the

existence of the partial derivatives, but the converse does not hold. So we have proved the traditional relations between them. In particular, we have proved that, for bivariate functions, if at least one of the partial derivatives is continuous, then Frechet's derivative exists.

Things get hairier when higher derivatives are needed. An example is Schwarz' theorem, which states that mixed second partial derivatives are equal (that is, the Hessian matrix is symmetric) under some continuity hypotheses. Our formalism makes it easy to state, while it would have been nearly impossible with the standard library. Indeed, proof terms would creep every time the `Derive` total function appears in the statement below, and they would be especially intricate since the derivatives only need to exist in a neighborhood of (x, y) and not on the whole $\mathbb{R}^2$ space.

```
Lemma Schwarz : ∀ f x y,
  locally_2d (fun u v ⇒
    ex_derive (fun z ⇒ f z v) u ∧
    ex_derive (fun z ⇒ f u z) v ∧
    ex_derive (fun z ⇒ Derive (fun t ⇒ f z t) v) u ∧
    ex_derive (fun z ⇒ Derive (fun t ⇒ f t z) u) v) x y →
  continuity_2d_pt (fun u v ⇒ Derive (fun z ⇒
                        Derive (fun t ⇒ f z t) v) u) x y →
  continuity_2d_pt (fun u v ⇒ Derive (fun z ⇒
                        Derive (fun t ⇒ f t z) u) v) x y →
  Derive (fun z ⇒ Derive (fun t ⇒ f z t) y) x =
  Derive (fun z ⇒ Derive (fun t ⇒ f t z) x) y.
```

We have not restricted ourselves to second partial derivatives; we have also proved theorems about higher ones. An instance of such theorems is the Taylor-Lagrange approximation. It states that the following equality holds under some hypotheses on the bivariate function f:

$$f(x', y') = \texttt{DL\_pol}(f, n, x, y, x' - x, y' - y) + O(\|(x', y') - (x, y)\|^{n+1})$$

with

$$\texttt{DL\_pol}(f, n, x, y, u, v) = \sum_{p=0}^{n} \frac{1}{p!} \left(\sum_{m=0}^{p} \binom{p}{m} \cdot \frac{\partial^p f}{\partial x^m \partial y^{p-m}}(x, y) \cdot u^m \cdot v^{p-m} \right).$$

Again stating such a theorem would have been impossible if not for the availability of total functions for defining partial derivatives. It would also have been cumbersome with Frechet's higher derivative, since the n-th such derivative at a given point is a multilinear map from $\mathbb{R}^n$ to $\mathbb{R}$.

5.2. Parametric Integrals

Once one starts to manipulate analysis formulas mixing integrals and derivatives, there quickly comes a time when one has to compute the derivative of an integral (or the integral of a derivative). The fundamental theorem of analysis is sufficient to take care of the simpler case:

$$\left(x \mapsto \int_a^x f(t)dt \right)' (x) = f(x).$$

Unfortunately, integrals are hardly that simple and one might well have to compute the derivative of the following expression:

$$x \mapsto \int_{a(x)}^{b(x)} f(x, t)dt.$$

Not only do the bounds depend on the derivation variable x, but the integrated expression does too. In that case, the ability to commute derivative and integral becomes crucial. Fortunately, our

formalization provides the necessary tools. For instance, it proves that, under sufficient hypotheses, the derivative of the expression above is equal to

$$\int_{a(x)}^{b(x)} \frac{\partial f}{\partial x}(x,t)dt - f(x,a(x)) \cdot a'(x) + f(x,b(x)) \cdot b'(x).$$

An example that heavily manipulates derivatives and integrals is the proof of the relation between the elliptic integral of the first and second kinds and the arithmetic-geometric mean [2]. This was proved in Coq thanks to some theorems from Coquelicot.

5.3. Generalized Limits

Since $\mathbb{R}$ is equipped with a linear order, people want to compute limits of functions not only at finite points, but also at $+\infty$ and $-\infty$. This kind of limits is unfortunately missing from the standard library of Coq. So we added some support for them in Coquelicot, in a way similar to what can be found for other proof assistants. Yet this was not entirely satisfactory, as mathematicians are accustomed to writing the following kind of equalities:

$$\lim_{t \to x}(f(t) + g(t)) = \lim_{t \to x} f(t) + \lim_{t \to x} g(t)$$

whether x is finite or not, and whether the actual limits are finite or not.

Nets and filters make short work of the various cases for x, but they do not really help for the limit values themselves. There are indeed 7 cases to consider (cases $+\infty - \infty$ and $-\infty + \infty$ being forbidden), which thus flood the user with theorems. Even accounting for commutativity and sign symmetries, there still remain 3 cases, and they come at the expense of user-friendliness since they require preparatory work from the user. Therefore, we wanted to devise a system that would encompass all these cases.

We have done so by defining limits as having values in the set $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. We also extended the addition to a function of type $\overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$ and proved that it was continuous over $\overline{\mathbb{R}}^2 \setminus \{(+\infty, -\infty), (-\infty, +\infty)\}$. As a consequence, the above equality about the sum of limits can now be written. Moreover, its proof is an immediate corollary of the continuity of addition. The other usual arithmetic operators are handled the same way by the library.

The intermediate value theorem is another example where its generalization to $\overline{\mathbb{R}}$ is definitely useful. Given an open interval (a, b), a function $f : \mathbb{R} \to \mathbb{R}$ continuous over (a, b), and a real y, this theorem provides a real x such that $f(x) = y$ when $\lim_{t \to a} f(t) < y < \lim_{t \to b} f(t)$. Note that this theorem holds whether the bounds a and b, and the two limits of f, are finite or not. It is thus important that the user be given only one single theorem rather than 16 variants. Moreover, the theorem proved in the Coquelicot library is stated as a function rather than with an existential quantifier, so that it can be used to invert any continuous function.

5.4. Asymptotic Behaviors

The library also provides some definitions for expressing when functions are asymptotically equivalent or when a function asymptotically dominates another. These properties are defined for functions from an arbitrary space T to $\mathbb{R}$, and filter F describes the limit point of T at which to compare the functions:

```
Definition is_domin (F : (T → Prop) → Prop) (f g : T → R) :=
  ∀ eps : posreal, F (fun x ⇒ Rabs (g x) ≤ eps * Rabs (f x)).
Definition is_equiv (F : (T → Prop) → Prop) (f g : T → R) :=
  is_domin F g (fun x ⇒ g x - f x).
```

Relation `is_domin` is proved to be a strict partial order, while `is_equiv` is an equivalence. The usual correspondences between them are also provided by the library. Moreover, given a function f, the set of functions `is_domin` (F, f) is a vector space of functions, while the equivalence

class of f is the affine space f + `is_domin` (F, f). These properties take care of addition and multiplication by a constant; the library also provides theorems for multiplication and division.

Finally, the whole point of knowing that functions are asymptotically equivalent is for proving limits. If two functions f and g are equivalent with respect to F and l is the limit (possibly infinite) of f, then l is also the limit of g:

```
Lemma filterlim_equiv : ∀ (f g : T → R) (l : Rbar),
  is_equiv F f g → filterlim f F (Rbar_locally l) →
  filterlim g F (Rbar_locally l).
```

5.5. Tactics for Automating Reasoning about Differentiability

When it comes to real analysis, some reasonings are mechanical, guided only by the structure of expressions, *e.g.* differentiability. It is thus important not only to provide tools that ease writing mathematics, but also tools that automate reasoning whenever possible.

For that purpose, the library provides the `auto_derive` tactic. It is meant to help proving statements of the form `is_derive` (f, x, l), that is, f has a derivative at point x and it is equal to l. Variants of this tactic take care of equivalent statements from the standard library, or statements where l is implicit. The tactics first reifies expression f, then performs symbolic differentiation on it, and finally compares the obtained derivative with l. Along the differentiation computations, it keeps track of all the side conditions needed for the final result to actually be a derivative of the original expression. These properties are left for the user to prove. For example, consider the following goal; it contains an uninterpreted function symbol $f : \mathbb{R} \to \mathbb{R}$, two real variables x and l, and a parametric integral:

```
Goal is_derive (fun t ⇒ f t + RInt (fun u ⇒ cos (t + u)) 0 1) x l.
```

Given this goal, the tactic requires the user to prove that f is differentiable at x, that the integral $\int_0^1 \cos(x'+u)du$ exists for any x' in some neighborhood of x, that the function $(x', u) \mapsto \sin(x'+u)$ is continuous at any point of $\{x\} \times [0, 1]$, and that the following equality holds:

$$f'(x) + \int_0^1 -\sin(x+u)du = l.$$

The tactic supports all the functions defined in the standard library. Through the declaration of type class instances, the user can extend it to support more differentiable functions. In fact, this is the mechanism used by the tactic itself, as its core only knows about basic arithmetic operations and integrals.

Coq's standard library provides a `reg` tactic that can be used for a similar purpose. It suffers from a few shortcomings though. First, it does not support integrals; it is not extensible to more functions either. Second, it is only suitable for proving differentiability, not for computing derivatives. Indeed, it tries to compute derivatives only for expressions of the form `derive_pt`, that is, expressions that contain a proof term. This means that the user already had to manually prove the property beforehand, which kind of defeats the point. The `reg` tactic has some other uses outside differentiability though, as it is also able to prove continuity properties.

6. Applications

Our definitions and lemmas cover a broad part of real analysis and we moreover claim they are user-friendly. To convince the reader, we now show three very different applications that demonstrate various features of our library.[7]

[7] These applications are available in the *examples* directory of the Coquelicot distribution.

6.1. Baccalaureate

The first example is the most basic of the three applications. Most 18-year old French students pass an exam called Baccalaureate which ends the high school and is required for attending the university. The idea was to try Coquelicot on the 2013 mathematics test of the scientific Baccalaureate. Is a Coq user able to prove in a short time what is expected from scientifically-inclined French students at 18?

6.1.1. The experiment. We have tested several exams from the previous years, but we wanted to experiment in real-life test conditions. Therefore, one of the authors went to the "Parc de Vilgénis" high school in Massy, France and took the 2013 test at the same time as the students. There was therefore no possible cheating: the library was already developed and it was tested as is during the four hours of the test. The first exercise was about probabilities and the third one about geometry, and are therefore out of the scope of the library. Let us focus on the two other exercices from the test [1].

6.1.2. Exercise 2. The second exercise was exactly real analysis. Some function was defined:

$$f(x) = \frac{a + b\ln(x)}{x}$$

and it had to be thoroughly studied: derivative, variation, intermediate value, several limits, and integral (between e^{-1} and 1).

Most of the demonstrations were done in a little more than 2 hours. They required about 300 lines of Coq. They proved the adequacy of the library for derivatives and integrals. Proving things in such a limited amount of time was quite a challenge and we succeeded in formally proving the expected answers. Note that the proofs were done using only the most basic tactics of Coq, for the sake of the experiment.

Some questions would not have been feasible using the standard library. Indeed, the students were to give the limits of f when x goes to zero and when x goes to infinity, neither of which can be stated using the standard Coq library.

An unexpected difficulty (that did not arise in the previous tests we trained on) was the use that year of right and left limits which were not formalized in Coquelicot at that time. Proving the limit of f when x goes to 0 requires x to be positive. The trick used during the test was to give the limit of $f(|x|)$ when x goes to zero, as $f(|x|)$ is defined for all nonzero reals. Therefore, the correct answer was formally proved, even if a twist had to be used.

6.1.3. Exercise 4. The fourth and last exercise was about a sequence $(u_n)_{n\in\mathbb{N}}$ defined by

$$u_0 = 2 \qquad \text{and} \qquad u_{n+1} = \frac{2}{3}u_n + \frac{1}{3}n + 1.$$

The questions were its variation, several inequalities, other equivalent formulas of the same sequence, its limit when n goes to infinity and a small study (value, limit) of

$$T_n = \frac{\sum_{k=0}^{n} u_k}{n^2}.$$

All the demonstrations were rather easy and done in one hour. They required about 140 lines of Coq. Our definitions and lemmas about generalized limits were the only features used. They shortened the proofs, but most of them could have been done using the standard library.

6.1.4. Conclusion. After the test, a meeting was organized with some teachers and the produced proofs and results were presented. They seemed interested and found the total functions for the limit and the derivative quite an asset. They also agreed that a use in class is still unrealistic due to the use of tactics and the very low granularity of the proofs for algebraic manipulations.

Nevertheless, this experiment shows that Coquelicot is able to cope with basic real analysis: it has the necessary definitions and lemmas, and its usability and efficiency have been demonstrated in a test with a limited time. The generalized limits (towards infinities) were entirely required, while our lack of left and right limits (at that time) could be bypassed. The final Coq file is about 500 lines long for 19 lemmas (exactly corresponding to the test questions). The only questions left were the three following ones:

- Using the given graph of the function, the students had to visually evaluate $f(1)$ and $f'(1)$.
- An imperative algorithm was given and the students were supposed to run it by hand. Even if it would have been possible to formalize and prove it in Coq, it was both uninteresting, time-consuming and not related to real analysis, so we left out this question.
- The values of u_1, u_2, u_3, u_4 had to be computed with a 10^{-2} accuracy. This kind of computation is out of the scope of our library, but could have been done using for example the `interval` tactic [25].

Therefore very few questions are out of the scope of Coquelicot, showing also that its breadth is at least comparable to that of the most scientifically-inclined 18-year old French students.

6.2. Bessel Functions

The second example illustrates the use of power series. We have defined Bessel functions as follows.

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{p=0}^{+\infty} \frac{(-1)^p}{p!(n+p)!} \left(\frac{x}{2}\right)^{2p}.$$

This is a well-known power series with numerous recurrence relations and it will show how easily one can manipulate power series with Coquelicot. The first question regarding a power series is its convergence radius. Here, it is the simplest case as it is infinite. We therefore state using our extended reals:

```
Lemma CV_Bessel1 : ∀ n : nat, CV_radius (Bessel1_seq n) = p_infty.
```

Using d'Alembert criterion, this is easy and requires less than 40 lines.

We can now formally prove various relations and properties of Bessel functions [21]:

$$\begin{aligned}
&\forall x \in \mathbb{R}, \quad J_{n+1}(x) + J_{n-1}(x) = \tfrac{2n}{x} J_n(x),\\
&\forall x \in \mathbb{R}, \quad J_{n+1}(x) = \tfrac{nJ_n(x)}{x} - J_n'(x),\\
&\forall x \in \mathbb{R}, \quad J_{n+1}(x) - J_{n-1}(x) = -2J_n'(x).
\end{aligned}$$

The first one only requires arithmetic operations on power series, but the other ones require Bessel functions to be differentiable. Of course, we also proved that our function is solution of the Bessel equation:

$$\forall x \in \mathbb{R},\ x^2 \cdot J_n''(x) + x \cdot J_n'(x) + \left(x^2 - n^2\right) J_n(x) = 0.$$

What is interesting here is the statement of this last theorem: in mathematics, we directly write J_n' and J_n''. Using Coq's standard library, it would have been very complex, as we have to give the differentiability proofs both for J_n and for J_n', which would have given an ugly and unreadable expression. In Coquelicot, we get the following statement:

```
Lemma Bessel1_correct : ∀ (n : nat) (x : R),
  x^2 * Derive_n (Bessel1 n) 2 x + x * Derive (Bessel1 n) x
          + (x^2 - (INR n)^2) * Bessel1 n x = 0.
```

where `Derive_n` is the n-th derivative of the function at a given point. Except for `INR` (which transforms a natural number into a real number), this is quite readable and close to the mathematical expression.

To formally prove the values of $J'_n(x)$ and $J''_n(x)$, we first proved the existence of a derivative, which is easy as both are power series with an infinite convergence radius. Then, we called the `auto_derive` tactic and proved the correspondence between two algebraic expressions.

At the end, we have a 300-line long file that defines and gives several properties of the Bessel function. The brevity of the file shows that power series are practical to use and that the library is user-friendly.

6.3. D'Alembert Formula

This last 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

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

with some initial conditions for $u(x,0)$ and $\frac{\partial u}{\partial t}(x,0)$.

To actually compute an approximation of the solution u, we choose 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 some similar formulas that depend on u_0 and u_1 for the initializations u_j^0 and u_j^1.

The full C program was verified for a finite rope [5], including both the rounding error and the method error. The proofs of that development were relying on two axioms: the existence of a solution to the partial differential equation and its regularity. Thanks to the Coquelicot library, we are now able to prove those assumptions.

For the existence, the mathematical proof is simpler than one could expect, as this equation has an analytical solution. More precisely, d'Alembert's formula

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

defines a function that is solution to the previous partial differential equation. We define α, β, and γ, as parts of this formula and we just have to derive them and check several equalities. This called for automation, and led to a reflexive Coq tactic for proving differentiability [22]. The tactic was limited: it could handle expressions with one variable only. As a consequence, human intervention was needed for differentiating under the integral sign of γ. 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. Now that we have got rid of proof terms in values,

the `auto_derive` tactic is able to differentiate below the integral sign and can thus be used for this application.

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. 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 order 3. Both were proved using the Taylor-Lagrange approximation theorem described in Section 5.1.

This application was the initial reason for the Coquelicot library: this property is not provable using the standard library. Therefore, we decided to build the necessary tools to tackle this problem: easy statements for derivatives and integrals, and automations. Consider for example the second partial derivative with respect to the first variable of γ, that is

$$\frac{\partial^2 \gamma}{\partial x^2}(x,t) = \frac{1}{2c}\int_0^t \frac{\partial f}{\partial x}(x + c(t-\tau), \tau) - \frac{\partial f}{\partial x}(x - c(t-\tau), \tau)\, d\tau.$$

It is a complex mix of integrals and derivatives that would have been a nightmare to write using the standard library. Using Coquelicot, the property is quite readable:

```
Definition gamma20 x t := 1/(2*c) * RInt
      (fun tau ⇒ Derive (fun u ⇒ f u tau) (x + c * (t - tau)) -
          Derive (fun u ⇒ f u tau) (x - c * (t - tau)))
      0 t.

Lemma gamma20_lim :
  ∀ x t, is_derive_n (fun u ⇒ gamma u t) 2 x (gamma20 x t).
```

Another fact is that this application requires definitions and lemmas that are not provided by the standard library: functions of two variables (continuity and so on) and parametric integrals.

This application is rather specific, and its success indicates that automations and user-friendly statements are fulfilled requirements, as proofs are quite short and as theorems are easy to read. With the other applications, the breadth of the library appears, from university level to power series and parametric integrals.

7. Conclusion

The Coquelicot library provides an extension of Coq's standard library for real analysis. It contains nearly 200 definitions, more than 1,000 lemmas, and about 20,000 lines of Coq proofs. Coquelicot provides homogeneous definitions and lemmas for limits, derivatives, integrals, series, and power series. Whenever different, these definitions are proved equivalent to those of the standard library, so that user developments can easily mix both libraries. Moreover, no new axioms were introduced.

For all of those notions of real analysis, total functions are available. These functions return the expected value in case of convergence and an arbitrary (extended) real number otherwise. By avoiding dependent types throughout the library, theorem statements become much easier to write than with the standard library. Indeed, hypotheses of convergence, differentiability, and so on, can now be split from the actual values. Without such a feature, it would be next to impossible to state theorems such as those about d'Alembert's formula. It also simplifies the proof process, as it makes rewriting rules usable in practice. This approach based on total functions is similar to what other

proof assistants achieve thanks to Hilbert's ε operator, but without having to assume the existence of such an operator in Coq.

These total functions are implemented by combining the limited principle of omniscience with the completeness axiom of the real numbers. For topological properties, the library relies on the pervasive use of filters. Arithmetic and order on extended real numbers allow to easily express theorems about limits and power series without having to separate finite and infinite cases. In addition to those basic features, we have also developed several extensions: differentiability of bivariate functions, parametric integrals, asymptotic behaviors, and a tactic for automating proofs of differentiability. Finally, a great care has been taken when naming and stating theorems, so as to avoid some of the most unnerving issues of the standard library.

In this paper, we have also presented a few applications that either motivated the library or were used to exercise it. Indeed, we consider important, when formalizing a library for mathematics, to keep applications in mind, so as to ensure it will actually be usable in practice.

To conclude, Coquelicot is a notable improvement over Coq's standard library and it is suitable for our current applications. Still, we wish to extend it further. First, a natural perspective is to define new elementary functions and prove their properties. It is feasible as is in a real-only setting, but it would be much more satisfying to have complex numbers for that purpose. Second, automations could be improved further, so as to handle predicates such as integrability and multivariate continuity. Finally, while a comprehensive support for parametric integrals is a non-negligible feature, integration is still lacking, especially when it comes to surface or path integration. For instance, proving a theorem such as Kelvin-Stokes' is currently unpractical.

Acknowledgements

The authors are grateful to Pierre Michalak and Évelyne Roudneff for allowing us to take the Baccalaureate exam in real-life conditions in a high school in Massy, and for organizing the meeting with interested teachers.

References

[1] Baccalauréat général, Série S, Mathématiques, Session 2013, June 2013. `http://eduscol.education.fr/prep-exam/sujets/13MASCOMLR1.pdf`.

[2] Yves Bertot. Proving the convergence of a sequence based on algebraic-geometric means to π, 2013. `http://www-sop.inria.fr/members/Yves.Bertot/proofs.html`.

[3] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.

[4] Frédéric Besson. Fast reflexive arithmetic tactics: the linear case and beyond. In *Proceedings of the International Conference on Types for proofs and programs (TYPES'06)*, volume 4502 of *Lecture Notes in Computer Science*, pages 48–62, Nottingham, UK, 2006.

[5] Sylvie Boldo, François Clément, Jean-Christophe Filliâtre, Micaela Mayero, Guillaume Melquiond, and Pierre Weis. Wave equation numerical resolution: a comprehensive mechanized proof of a C program. *Journal of Automated Reasoning*, 50(4):423–456, April 2013.

[6] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Improving real analysis in Coq: a user-friendly approach to integrals and derivatives. In Chris Hawblitzel and Dale Miller, editors, *Proceedings of the 2nd International Conference on Certified Programs and Proofs*, volume 7679 of *Lecture Notes in Computer Science*, pages 289–304, Kyoto, Japan, 2012.

[7] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Formalization of real analysis: A survey of proof assistants and libraries. 2013. Submitted. `http://hal.inria.fr/hal-00806920`.

[8] Cyril Cohen. Reasoning about big enough numbers in Coq. In *Proceedings of the 4th Coq Workshop*, Princeton, NJ, USA, 2012.

[9] Luis Cruz-Filipe, Herman Geuvers, and Freek Wiedijk. C-CoRN: the constructive Coq repository at Nijmegen. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Proceedings of the 3rd International Conference of Mathematical Knowledge Management*, volume 3119 of *Lecture Notes in Computer Science*, pages 88–103, 2004.

[10] Marc Daumas, David Lester, and César Muñoz. Verified real number calculations: A library for interval arithmetic. *IEEE Transactions on Computers*, 58(2):226–237, 2009.

[11] Bruno Dutertre. Elements of mathematical analysis in PVS. In J. von Wright, J. Grundy, , and J. Harrison, editors, *Proceedings of the 9th International Conference Theorem Proving in Higher Order Logics*, volume 1125 of *Lecture Notes in Computer Science*, pages 141–156, 1996.

[12] Jacques Fleuriot. On the mechanization of real analysis in Isabelle/HOL. In Mark Aagaard and John Harrison, editors, *Proceeding of the 13th International Conference of Theorem Proving in Higher Order Logics*, volume 1869 of *Lecture Notes in Computer Science*, pages 145–161, 2000.

[13] Ruben Gamboa and Matt Kaufmann. Nonstandard analysis in ACL2. *Journal of Automated Reasoning*, 27(4):323–351, November 2001.

[14] Herman Geuvers and Milad Niqui. Constructive reals in Coq: Axioms and categoricity. In Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack, editors, *Selected Papers of the International Workshop on Types for Proofs and Programs (TYPES 2000)*, volume 2277 of *Lecture Notes in Computer Science*, pages 79–95, 2002.

[15] John Harrison. Constructing the real numbers in HOL. *Formal Methods in System Design*, 5(1–2):35–59, July 1994.

[16] John Harrison. HOL Light: An overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)*, volume 5674 of *Lecture Notes in Computer Science*, pages 60–66, Munich, Germany, 2009.

[17] John Harrison. The HOL Light theory of Euclidean space. *Journal of Automated Reasoning*, 50:173–190, 2013.

[18] Johannes Hölzl, Fabian Immler, and Brian Huffman. Type classes and filters for mathematical analysis in Isabelle/HOL. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Proceedings of the 4th International Conference on Interactive Theorem Proving (ITP)*, volume 7998 of *Lecture Notes in Computer Science*, pages 279–294, Rennes, France, 2013.

[19] Cezary Kaliszyk and Russell O'Connor. Computing with classical real numbers. *Journal of Formalized Reasoning*, 2(1):27–39, 2009.

[20] Robbert Krebbers and Bas Spitters. Type classes for efficient exact real arithmetic in Coq. *Logical Methods in Computer Science*, 9(1:1):1–27, 2013.

[21] Catherine Lelay. A New Formalization of Power Series in Coq. In *5th Coq Workshop*, pages 1–2, Rennes, France, July 2013. `http://coq.inria.fr/coq-workshop/2013#Lelay`.

[22] Catherine Lelay and Guillaume Melquiond. Différentiabilité et intégrabilité en Coq. Application à la formule de d'Alembert. In *23èmes Journées Francophones des Langages Applicatifs*, pages 119–133, Carnac, France, 2012.

[23] Micaela Mayero. *Formalisation et automatisation de preuves en analyses réelle et numérique*. PhD thesis, Université Paris VI, décembre 2001.

[24] Sean McLaughlin and John Harrison. A proof-producing decision procedure for real arithmetic. In Robert Nieuwenhuis, editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE-20)*, volume 3632 of *Lecture Notes in Computer Science*, pages

295–314, Tallinn, Estonia, 2005.

[25] Guillaume Melquiond. Proving bounds on real-valued functions with computations. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 2–17, Sydney, Australia, 2008.

[26] César Muñoz and Anthony Narkawicz. Formalization of a representation of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, pages 1–46, 2012.

[27] Adam Naumowicz and Artur Korniłowicz. A brief overview of Mizar. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proceedings of the 22th International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 67–72, 2009.

[28] Sam Owre, John Rushby, and Natarajan Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992.

[29] Russell O'Connor. A monadic, functional implementation of real numbers. *Mathematical Structures in Computer Science*, 17:129–159, 1 2007.

[30] Russell O'Connor and Bas Spitters. A computer-verified monadic functional implementation of the integral. *Theoretical Computer Science*, 411(37):3386–3402, 2010.

[31] Loïc Pottier. Connecting Gröbner bases programs with Coq to do proofs in algebra, geometry and arithmetics. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Knowledge Exchange: Automated Provers and Proof Assistants*, CEUR Workshop Proceedings, pages 67–76, Doha, Qatar, 2008.

[32] John Rushby, Sam Owre, and Natarajan Shankar. Subtypes for specifications: Predicate subtyping in PVS. *IEEE Transactions on Software Engineering*, 24(9):709–720, sep 1998.

[33] Andrzej Trybulec. Some features of the Mizar language. In *Proceedings of the ESPRIT Workshop*, Torino, Italy, 1993.

[34] Andrzej Trybulec. Non negative real numbers. Part I. *Journal of Formalized Mathematics*, Addenda, 1998.

Sylvie Boldo
Inria Saclay - Île-de-France
PCRI - Batiment 650
Université Paris-Sud
91405 Orsay cedex, France
e-mail: sylvie.boldo@inria.fr

Catherine Lelay
Inria Saclay - Île-de-France
PCRI - Batiment 650
Université Paris-Sud
91405 Orsay cedex, France
e-mail: catherine.lelay@inria.fr

Guillaume Melquiond
Inria Saclay - Île-de-France
PCRI - Batiment 650
Université Paris-Sud
91405 Orsay cedex, France
e-mail: guillaume.melquiond@inria.fr