Formalization of Real Analysis: A Survey of Proof Assistants and Libraries

Sylvie Boldo, Catherine Lelay, Guillaume Melquiond

Under consideration for publication in Math. Struct. in Comp. Science

Formalization of Real Analysis: A Survey of Proof Assistants and Libraries*

SYLVIE BOLDO

CATHERINE LELAY

GUILLAUME MELQUIOND

Inria,
LRI, bâtiment 650,
Université Paris-Sud,
F-91405 Orsay Cedex, France

Received 25 March 2013; Revised 20 December 2013

In the recent years, numerous proof systems have improved enough to be used for formally verifying non-trivial mathematical results. They, however, have different purposes and it is not always easy to choose which one is adapted to undertake a formalization effort. In this survey, we focus on properties related to real analysis: real numbers, arithmetic operators, limits, differentiability, integrability, and so on. We have chosen to look into the formalizations provided in standard by the following systems: Coq, HOL4, HOL Light, Isabelle/HOL, Mizar, ProofPower-HOL, and PVS. We have also accounted for large developments that play a similar role or extend standard libraries: ACL2(r) for ACL2, C-CoRN/MathClasses for Coq, and the NASA PVS library. This survey presents how real numbers have been defined in these various provers and how the notions of real analysis described above have been formalized. We also look at the methods of automation these systems provide for real analysis.
Keywords: Formal proof systems, real arithmetic, real analysis, libraries.

1. Introduction

Rational numbers have been used for millennia to compute areas, taxes, etc. Calculus has been developed in the 18th century and now all students are used to knowing and computing with 3, $\frac{1}{7}$, $\sqrt{13}$, $\exp(1)$, and π. Calculus and computations are even seen as a way to select the assumed "best" students. Real analysis is indeed a good test case for students: knowing definitions and theorems, they are assessed for reasoning and correctly applying theorems. When it comes to proof assistants, the situation is the opposite: they are designed for reasoning, but are to be assessed for definitions and theorems about real analysis. Our objective is similar to that of Wiedijk (2006): he compared seventeen

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

provers by showing how to prove that $\sqrt{2}$ is irrational. (Note that this lemma can be stated using only integers and thus does not tell much about support for real analysis.) The appeal of that work is that it collects proof written by power users of those formal systems. Wiedijk (2009) then compared libraries: that latter work presents statistics about the whole standard libraries of some provers, but it does not look at the actual definitions and theorems.

Our goal is to compare the formal systems on their abilities to prove mathematical properties of real analysis and how much help they provide to the user. The aim of this survey is to give the reader an overview of the available systems, their libraries and tools for formal reasoning about real analysis. This comparison is needed since many different design choices have been made. That way, a user accustomed to formal methods may choose the most appropriate system for proving a real analysis result. No such survey has been written before, and even the available literature is sometimes insufficient to compare libraries and definitions.

Formal proof assistants are quite recent in computer science history: the first influential system was AutoMath in 1967, though there were some earlier works (Guard et al. 1969). Then came Boyer-Moore (1971), LCF (1972), Mizar (1973), and many more systems. Real number formalizations soon followed (Jutting 1977) and research is ongoing. Our goal here is not to survey all the provers. For the sake of significance, we restrict ourselves to systems that stood the test of time. Moreover they have to come with a standard library for real arithmetic. They also have to provide support for real analysis, and not just a construction of real numbers. The standard libraries we have picked out are those of Mizar, PVS, HOL4, HOL Light, Isabelle/HOL, and Coq. We have also accounted for large developments that play a role similar to standard libraries: ACL2(r) for ACL2, C-CoRN/MathClasses for Coq, and the NASA PVS library. Another choice of ours is to focus on real analysis and to ignore probabilities and most of measure theory.

This survey is organized as follows: Section 2 describes the various proof assistants and libraries we will focus on. Section 3 shows the various formalizations of the set $\mathbb{R}$ of real numbers. Section 4 outlines the choices they made for the usual definitions found in real analysis. Available methods of automation are presented in Section 5.

2. Proof Assistants and Libraries

This section presents the proof assistants and the libraries we will focus on in the rest of this survey. Proof assistants may share several characteristics.

First, the input language may be either declarative or procedural (Harrison 1996). In a declarative system, the user makes explicit the intermediate states while the proof steps are implicit. On the contrary, the procedural style uses tactic-based languages that makes explicit the proof steps while intermediate states are implicit.

Second, some proof assistants follow the LCF approach: they rely on a small kernel written in a ML-like programming language, only basic inference rules are allowed, and the user may write theorem-proving tactics. Other approaches will be detailed when needed.

2.1. *Mizar*

Mizar* is a formal system of general applicability, based on classical logic and the Jaskowski system of natural deduction (Trybulec 1993, Naumowicz and Korniłowicz 2009). A proof is a Mizar script, that is checked by the system.

The primary goal is that the proofs be close to the mathematical vernacular, so that mathematicians may easily use it. The language is thus purely declarative. 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. Mizar is also used for teaching purposes.

The logic is based on the axioms of the Tarski-Grothendieck set theory (an extension of the Zermelo-Fraenkel set theory). All objects are sets, and specific ones can be called integers or reals. 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. The definition of real numbers is described in Section 3.2.5.

2.2. *ACL2(r)*

ACL2† is a system based on a first-order logic with an induction rule of inference (Kaufmann et al. 2000). Among the systems we consider here, ACL2 is the only one in a first-order setting. It is widely and industrially used (Moore et al. 1998) as it is robust and has good prover heuristics.

ACL2 is written in Common Lisp. The user only submits definitions and theorems and the system tries to infer proofs. In case of proof failure, one can then add lemmas such as rewriting rules, give hints such as disabling a rule or instantiating a theorem. Quantifiers in formulas are not directly supported, but ACL2 offers some ways for simulating them by skolemization. After developing a theory in ACL2, the user may execute it: for example after formalizing a microprocessor, it is possible to simulate a run of it.

ACL2(r) is an extension of ACL2 that offers support for reasoning about irrational and complex numbers. It modifies the ACL2 logic by introducing notions from non-standard analysis (Gamboa and Kaufmann 2001). Its real numbers are described in Section 3.3.1.

2.3. *PVS and NASA Library*

PVS‡ is a formal system based on classical higher-order logic (Owre et al. 1992). Contrarily to the LCF approach to provers, PVS has no small kernel, but a large monolithic system containing the checker and many methods of automation. It heavily uses predicate subtypes and dependent types (Rushby et al. 1998); the TCCs (type-correctness conditions) are proof obligations generated by the PVS typechecker, for example to prevent division by zero.

PVS is primarily written in Common Lisp and the user interface is built on Emacs.

*`http://mizar.org/`
†`http://www.cs.utexas.edu/users/moore/acl2`
‡`http://pvs.csl.sri.com/`

There is a user-written file for the specification which is typechecked. The proofs are done interactively and stored in a separate file that describes a sequent-based proof tree. There is also a script mode called ProofLite that can be used to get an offline behavior similar to other systems. Usability is polished with efficient decision procedures and a large use of typechecking information for the proofs. The formalization of real numbers is described in Section 3.1.1. A large PVS library maintained by the NASA[§] provides a formalization of real analysis.

2.4. *HOL Light*

HOL Light[¶] is a formal system based on classical higher-order logic with axioms of infinity, extensionality, and choice in the form of Hilbert's ϵ operator (Harrison 2009). It follows the LCF approach with a small kernel written in OCaml (Harrison 2006). The basic inference rules may be composed and methods of automation are available. The user may also program its own methods of automation. HOL Light was almost entirely written by Harrison. However, it builds on earlier versions of HOL, notably the original work by Gordon and Melham (1993) and the improved implementation by Slind.

Its formalization of real analysis originated in HOL88 (Harrison 1998) and large parts of it were ported to HOL Light, HOL4, and Isabelle/HOL. Motivated by the Flyspeck project (Hales 2012), the HOL Light library of real analysis was later generalized into a formalization of Euclidean spaces while keeping the same construction of real numbers (Harrison 2013). This survey only covers this latter formalization. Details about the earlier formalization can be inferred from the description of the HOL4 library of real analysis as it has not diverged much. The real numbers of HOL Light are described in Section 3.2.1.

2.5. *HOL4*

HOL4[‖] is the fourth version of HOL. It follows the LCF approach with a small kernel written in SML. It builds on HOL98 but also on HOL Light ideas and tools. The logic is also the same (HOL4 development team 2012). External programs such as SMT or BDD engines can be called using an oracle mechanism. Its real numbers are described in Section 3.2.6.

2.6. *ProofPower-HOL*

ProofPower-HOL[**] is another heir to HOL. It follows the HOL approach, is written in SML, and shares the same logic as HOL Light and HOL4. A distinctive point is that ProofPower-HOL also provides support for specifications and proofs in Z using a semantic

[§] `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/`
[¶] `http://www.cl.cam.ac.uk/users/jrh/hol-light/index.html`
[‖] `http://hol.sourceforge.net/`
[**] `http://www.lemma-one.com/ProofPower/index/`

embedding of Z into HOL (Arthan and King 1996). It also contains a tool supporting refinement of Z to the SPARK subset of Ada.

The real analysis library was developed since 2001 in order to test the formalization of real numbers, described in Section 3.2.7. Now, the real analysis library aims at proving the corresponding theorems among the 100 famous theorems of Wiedijk's webpage.

2.7. *Isabelle/HOL*

Isabelle/HOL[††] is a formal system based on higher order logic with a general natural deduction environment (Nipkow et al. 2002). Isar is a specific extension for declarative style (Wenzel 2002, Wenzel and Wiedijk 2002). Isabelle uses several generic proof tools (higher-order rewriting, classical proof search, arithmetic, etc) and implements a number of derived specification mechanisms on top of the existing kernel (inductive sets and data-types, recursive functions, extensible records etc). Like other LCF-style provers, all proofs expand to primitive inferences in the kernel. Theory and proof developments are highly interactive, both for unstructured tactic scripts and structured Isar proof texts. Its real numbers are described in Section 3.2.2.

2.8. *Coq: Standard Library and C-CoRN/MathClasses*

The formal language of Coq[‡‡] is based on the Calculus of Inductive Constructions which combines both a higher-order logic and a richly-typed functional programming language (Bertot and Castéran 2004). Programs can be extracted from proofs to external programming languages like OCaml, Haskell, or Scheme. As a proof development system, Coq provides interactive proof methods, decision and semi-decision algorithms, and a tactic language for letting the user define new proof methods.

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. The standard library reals are axiomatic and described in Section 3.1.2.

Recent developments in Nijmegen led to a different library called C-CoRN (Cruz-Filipe et al. 2004) and its successor MathClasses (Spitters and van der Weegen 2011, Krebbers and Spitters 2013). The main idea is to provide Bishop-like constructive mathematics. These libraries are described in Section 3.2.3.

Another important library comes from the Mathematical Components project (Garillot et al. 2009). It is a comprehensive formalization when it comes to number theory and algebraic structures, but real analysis is in its infancy, so we leave this library out of this survey.

[††] `http://isabelle.in.tum.de/`
[‡‡] `http://coq.inria.fr/`

3. Formalizations of Real Numbers

Before even tackling the topic of real analysis, one has to understand how real numbers are represented in the various systems and libraries. They have chosen vastly different approaches and we will classify them into three categories. First, Section 3.1 presents formalizations that characterize real numbers as a given set with specific operations and properties. In Section 3.2, real numbers are actually built upon rational numbers. Finally, Section 3.3 regroups formalizations that are not based on real numbers but rather on hyper-reals. Whenever relevant, we also mention whether analysis is built directly on those sets or on higher-level concepts like topological spaces.

3.1. *Axiomatized Real Numbers*

The PVS system and the standard library of Coq both postulate real numbers as a complete Archimedean field. This leads to short and intuitive formalizations, but there is always a slight doubt on whether the additional axioms might have introduced an inconsistency with the native axioms of these systems.

3.1.1. *PVS* Thanks to its pervasive support for subtyping, PVS takes a top-down approach. Instead of starting from natural numbers and building more and more complicated types on top of them (as is done in other systems), PVS starts from a `number` type that is a superset of all numerals (Owre and Shankar 2003). This set encompasses integers, rationals, real numbers. Note that integers are not formalized in PVS: they are the native integers from the underlying Lisp system which runs PVS. In particular, any Lisp computation on integers plays the same role as an axiom in PVS.

PVS then defines a subtype `number_field` of `number` that provides the field operators and axiomatizes their properties. Real numbers, complex numbers, hyper-real numbers, etc, will all be subtypes of `number_field`. Below is a selection of some theory declarations that start this hierarchy. Of particular interest is that the division is an operation between a number and a nonzero number. It means that, whenever the user writes a division, PVS adds an implicit precondition to the formula that requires a proof that the denominator is nonzero.

```
number: NONEMPTY_TYPE
number_field: NONEMPTY_TYPE FROM number
nonzero_number: NONEMPTY_TYPE = {r: number_field | r /= 0} CONTAINING 1
/: [number_field, nonzero_number -> number_field]
n0x: VAR nonzero_number
inverse_mult: AXIOM n0x * (1/n0x) = 1
```

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.

Real numbers are then defined as a subtype of `number_field` that is closed with respect

to the field operations and that provides a total order compatible with them. Below is an excerpt of the theories.

```
real: NONEMPTY_TYPE FROM number_field
nonzero_real: NONEMPTY_TYPE = {r: real | r /= 0} CONTAINING 1
nzreal_is_nznum: JUDGEMENT nonzero_real SUBTYPE_OF nonzero_number
x, y: VAR real
n0z: VAR nonzero_real
closed_divides: AXIOM real_pred(x / n0z)
<(x, y):  bool
posreal_add_closed: POSTULATE x > 0 AND y > 0 IMPLIES x + y > 0
trichotomy: POSTULATE x > 0 OR x = 0 OR 0 > x
```

Note that the `POSTULATE` keyword means that, while they theoretically are axioms, PVS decision procedures are able to prove these properties. This means that, due to their implementation, the decision procedures come with their own sets of axioms regarding real numbers; this is similar to the $0 \neq 1$ case above. As a consequence, it is the combination of the explicit axioms from theories and the implicit ones from decision procedures that defines what real numbers are in PVS.

Now that real numbers are 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 (Dutertre 1996).

```
S: VAR (nonempty?[real])
upper_bound?(x, S): bool = FORALL (s: (S)): s <= x
least_upper_bound?(x, S): bool = upper_bound?(x, S) AND
  FORALL y: upper_bound?(y, S) IMPLIES (x <= y)
real_complete: AXIOM FORALL S:
  (EXISTS y: upper_bound?(y, S)) IMPLIES
  (EXISTS y: least_upper_bound?(y, S))
```

Finally, PVS defines rational numbers as a subtype of reals, integers as a subtype of rationals, and so on. As with $0 \neq 1$, the fact that literal constants 0, 1, 2, etc, are members of all these types and different from each others is hardcoded in the system. Note that the top-down approach does not preclude adding supersets above reals (*e.g.* complex numbers). While reals are already a subset of `number_field`, subtyping judgments can be added to make them subsets of other sets.

3.1.2. *Coq standard library* Contrarily to PVS, Coq does not have a native notion of subtypes. Thus, while the axioms are similar, there are numerous technical differences. First of all, natural numbers and real numbers are separate types. Natural numbers are defined as an inductive type (unary representation), while reals are purely axiomatized (Mayero 2001).

```
Parameter R : Set.                (* reals *)
Parameter R0 : R.                 (* 0 *)
Parameter Rplus : R -> R -> R.    (* + *)
Parameter Rlt : R -> R -> Prop.   (* < *)
```

```
Axiom R1_neq_R0 : 1 <> 0.
Axiom Rplus_comm : forall r1 r2:R, r1 + r2 = r2 + r1.
Axiom Rplus_lt_compat_l : forall r r1 r2:R, r1 < r2 -> r + r1 < r + r2.
```

Another difference is that the operations are total functions. In particular, the multiplicative inverse is defined even for zero, though none of the axioms gives any clue to what the actual result might be. For instance, the axiom below states the equality $x^{-1}x = 1$ under the hypothesis $x \neq 0$ only. In fact, one can prove in Coq that $0^{-1}0 = 0$, since 0^{-1} is a real and 0 is right-absorbing for multiplication. `R` has no meadow structure though, since $0^{-1} = 0$ is not provable.

```
Parameter Rinv : R -> R. (* inverse *)
Axiom Rinv_l : forall r:R, r <> 0 -> Rinv r * r = 1.
```

As in PVS, completeness of the real numbers in Coq is postulated as the existence of the least upper bound:

```
Definition is_upper_bound (E:R->Prop) (m:R) :=
  forall x:R, E x -> x <= m.
Definition is_lub (E:R->Prop) (m:R) :=
  is_upper_bound E m /\ (forall b:R, is_upper_bound E b -> m <= b).
Axiom completeness: forall E:R->Prop,
  (exists m:R, is_upper_bound E m) ->
  (exists x:R, E x) -> { m:R | is_lub E m }.
```

In the axiom above, the Coq notation `{m|L}` means $\exists m, L$, as would `exists m,L`. This duplication is due to the intuitionistic setting of Coq logic: `{m|L}` is a stronger form of the existential quantifier, equivalent to the one that appears in the other systems, *e.g.* HOL systems. Similarly, both `P\/Q` and `{P}+{Q}` are disjunctions in Coq, the second one being equivalent to the disjunction found in the other systems. It appears in the axiom that states that the comparison between two reals is decidable:

```
Axiom total_order_T:
  forall r1 r2:R, {r1 < r2} + {r1 = r2} + {r1 > r2}.
```

Finally, the existence of a function akin to the integer part is postulated, which gives the Archimedean property.

3.2. *Constructed Real Numbers*

This section details some formalizations that construct real numbers from Cauchy sequences or Dedekind cuts. An important point is that these formalizations do not introduce any new axioms. Instead, a set of numbers is built then proved to have the desirable properties of real arithmetic. Obviously, these constructions still depend on the axioms of the logic systems, so it might not be possible to transport a construction from one system to another.

3.2.1. *HOL Light* This formalization originated in HOL and then served as the basis for the standard library of HOL Light (Harrison 1998). Rather than using fully-featured sequences of rational numbers, it is based on nearly-additive sequences of natural numbers.

The predicate `is_nadd` characterizing such a sequence `x` is given below, with `dist` the function returning the distance between two natural numbers. (In HOL-like languages, `?` denotes the existential quantifier, while `!` is the universal one.)

```
let is_nadd = new_definition
  `is_nadd x <=> (?B. !m n. dist(m*x(n), n*x(m)) <= B * (m + n))`;;
```

In truth, this predicate characterizes nearly-multiplicative sequences, which is equivalent. It amounts to saying that the sequence x_n/n has a limit and this limit ℓ satisfies $\forall n,\ |x_n/n - \ell| < B/n$. This allows one to construct nonnegative real numbers from natural numbers only. Order `<<=` is defined as follows.

```
let nadd_le = new_definition
  `x <<= y <=> ?B. !n. x(n) <= y(n) + B`;;
```

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.

The addition of two nearly-additive sequences is performed pointwise, while the multiplication is the composition of sequences. They are shown to be commutative, associative, and one can build a multiplicative inverse with the expected properties. The comparison and arithmetic operations are then proved to be compatible with the following equivalence relation `===`:

```
let nadd_eq = new_definition
  `x === y <=> ?B. !n. dist(x(n), y(n)) <= B`;;
```

Thanks to this equivalence relation, the set of nonnegative real numbers is then defined as a quotient type `hreal`. Finally, the whole real line is built the same way one usually builds signed integers from natural numbers. Indeed, the formalization defines type `real` as the quotient of `hreal` pairs by the following equivalence relation:

```
let treal_eq = new_definition
  `(x1,y1) treal_eq (x2,y2) <=> (x1 + y2 = x2 + y1)`;;
```

3.2.2. *Isabelle/HOL* While real analysis in Isabelle/HOL is made of libraries ported from HOL Light, this is not the case for the basic construction of real numbers. Indeed, the formalization relies on sequences of rational numbers rather than nearly-additive sequence of natural numbers. Note that this is not the first construction of real numbers to be part of Isabelle/HOL: Fleuriot (2000) had originally used Dedekind cuts, but the Cauchy-based construction has now taken over entirely. There is not even a correspondence lemma between both formalizations, so the former Dedekind-based construction will not be detailed any further.

The Isabelle/HOL definitions below define what a Cauchy sequence is and what it means for a sequence to *vanish*, that is, to tend to zero.

```
definition cauchy :: "(nat => rat) => bool"
  where "cauchy X <-> (∀r>0. ∃k. ∀m≥k. ∀n≥k. |X m - X n| < r)"
definition vanishes :: "(nat => rat) => bool"
  where "vanishes X = (∀r>0. ∃k. ∀n≥k. |X n| < r)"
```

The formalization then defines a relation between Cauchy sequences which difference vanishes and proves that it is an equivalence. The relation is then used to create a quotient type `real`.

```
definition realrel :: "((nat => rat) × (nat => rat)) set"
  where "realrel = {(X, Y).
    cauchy X ∧ cauchy Y ∧ vanishes (λn. X n - Y n)}"
lemma equiv_realrel: "equiv {X. cauchy X} realrel"
typedef real = "{X. cauchy X} // realrel"
```

Moreover, as the definition of quotient in Isabelle/HOL is

```
definition quotient :: "'a set => ('a × 'a) set => 'a set set"
  where "A//r =
    (\<Union>x ∈ A. {r``{x}})"  -- {* set of equiv classes *}
```

with `{r``{x}}` the set of elements in relation with `x`, a real number is by definition the equivalence class of a given Cauchy sequence.

As already seen for HOL Light, arithmetic operations are defined on sequences and ported to the quotient type, which is then proved to be an Archimedean field with the least upper bound property.

3.2.3. *C-CoRN/MathClasses* The C-CoRN and MathClasses libraries for Coq take an approach different from all the other formalizations presented in this survey. Indeed, real analysis is formalized in the setting of constructive mathematics rather than classical logic. As such, they intrinsically rely on the intuitionistic logic of Coq. Combined with the extraction mechanism of this system, it means that, from an axiom-free proof of a property $\forall x \exists y\ R(x, y)$ for some relation R, one gets a function f such that $\forall x\ R(x, f(x))$. Note that this is not just an application of the axiom of choice; f can effectively compute a result for any input. Its actual efficiency depends on the proof though.

Constructive mathematics come with some hurdles and they are reflected in those Coq developments. Indeed, proofs require a much stricter discipline and might become more convoluted, as the tricks of the trade are no longer available. For instance, excluded middle, axiom of choice, equality between real numbers, and so on, are mostly out of reach. Some notions of analysis are also reworded or simply nonexistent, *e.g.* discontinuous yet total functions.

We first present C-CoRN as it is the original development. As the formalizations above, it constructs real numbers on top of Cauchy sequences (Geuvers and Niqui 2002). Coq, contrarily to HOL-style provers, has poor support for quotient types though. In particular, its logic causes the equality on elements of such types to be useless. As a consequence, the whole C-CoRN formalization is built on the notion of *setoid*, that is, a carrier set, a relation between elements of this set, and some proofs that the relation is an equivalence relation.

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 the signature `CReals` below (slightly simplified):

```
Record CReals : Type := {
  carrier :> COrdField;
  limit : CauchySeq carrier -> carrier;
  ax_Lim : forall s : CauchySeq carrier, SeqLimit s (limit s);
  ax_Arch : forall x : carrier, {n : N | x <= nring n} }.
```

The first field of the record states that the carrier type used for constructive reals is an ordered field. The second and third fields state that there exists a function from Cauchy sequences to the ordered field, and that this function computes the limit. The last field states that the carrier type is also Archimedean. The formalization then simply postulates the existence of a set `IR` with these properties:

```
Axiom IR : CReals.
```

A separate development explains how to build a complete metric space from a metric space, which gives a justification for the above postulate. This completion is done through the use of *regular* functions: given a positive rational number, they return an element of the metric space that is close enough to the value they are supposed to represent. As can be seen from the definition below, this property is related to Cauchy's criterion of convergence.

```
Definition is_RegularFunction (x:QposInf -> X) : Prop :=
  forall (e1 e2:Qpos), ball (m:=X) (e1+e2) (x e1) (x e2).
```

Finally, C-CoRN builds the setoid `Q` of rational numbers, proves it is a metric space, completes it, and proves that it is isomorphic to the axiomatized set `IR`.

```
Definition CR := Complete Q_as_MetricSpace.
Definition CRasCReals : CReals := ... CR ...
Lemma CRIR_iso : Isomorphism CRasCReals IR.
```

The MathClasses library is based on the completion monad $\mathfrak{C}$ defined by O'Connor (2007). It is a monad on the category of metric spaces and uniformly continuous maps between them. $\mathbb{R}$ is then defined as $\mathfrak{C}(\mathbb{Q})$. A real number x (seen as a regular function) is defined as being nonnegative when

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

The order $x \leq_{\mathbb{R}} y$ is then defined as $y - x$ nonnegative. Transcendental functions can be defined from $\mathbb{Q}$ to $\mathbb{R}$ and then lifted to functions from $\mathbb{R}$ to $\mathbb{R}$ (O'Connor 2008). Clever techniques for compression and range reduction are used to make computations efficient.

Abstract interfaces are heavily used to ease statements and proofs (Krebbers and Spitters 2013). Thanks to type classes, the algebraic and order hierarchies (setoid, group, ring, and so on) easily benefit from inheritance.

Finally, C-CoRN and Coq's standard library are not entirely unrelated: Kaliszyk and O'Connor (2009) provide an equivalence lemma between both libraries, under the axioms of the standard library.

3.2.4. *NASA PVS Library* In addition to its axiomatization of $\mathbb{R}$, PVS also comes with a development of constructive reals by Lester (2008). They are based on a stream of digits that gives Cauchy sequences.

```
x: VAR real
p: VAR nat
c: VAR [nat->int]
cauchy_prop(x, c): bool  =
  (FORALL p: c(p)-1 < x*2^p AND x*2^p < c(p)+1)
```

Then each operation is defined: addition, subtraction, multiplication, inverse, division, and power series. All the correctness lemmas relate the Cauchy implementation with the main implementation. Their statements look as follows.

```
x,y : VAR real
cx, cy: VAR cauchy_real
mul_lemma: LEMMA cauchy_prop(x,cx) AND cauchy_prop(y,cy)
                 => cauchy_prop(x*y, cauchy_mul(cx,cy))
```

3.2.5. *Mizar* While originally an axiomatization, the formalization of real numbers in Mizar was later changed to a construction based on Dedekind cuts (Trybulec 1998). Since Mizar is based on set theory, Dedekind cuts easily fit in this system. The hierarchy starts by defining the natural numbers and the nonnegative rational numbers `RAT+` on top of them. Then Dedekind cuts are formalized with their traditional definition:

```
func DEDEKIND_CUTS -> Subset-Family of RAT+ equals
  { A where A is Subset of RAT+ :
    for r being Element of RAT+ st r in A holds
    ( ( for s being Element of RAT+ st s <=' r holds s in A ) &
      ex s being Element of RAT+ st ( s in A & r < s ) )
  } \ {RAT+};
```

which translates to saying that A is a Dedekind cut if it is a strict subset of $\mathbb{Q}^+ \cup \{0\}$ such that

$$\forall r \in A,\ (\forall s \in \mathbb{Q}^+ \cup \{0\},\ s \leq r \Rightarrow s \in A) \wedge (\exists s \in \mathbb{Q}^+ \cup \{0\},\ r < s \wedge s \in A).$$

Although the empty set is usually excluded from Dedekind cuts, this is not the case in Mizar. The reason is that the empty set happens to represent 0 in Mizar, so it will not cause any issue in the following.

The nonnegative real numbers then follow by taking the union of nonnegative rational numbers and Dedekind cuts. As a consequence, the injection from nonnegative rational numbers to real numbers is just the identity function. Note that Dedekind cuts represent both rational and irrational numbers, so one would end up with every rational number being represented by two sets. Therefore, to obtain a unique representation for rational numbers, the definition of nonnegative real numbers removes rational Dedekind cuts from this union:

```
func REAL+ -> set equals (RAT+ \/ DEDEKIND_CUTS)
```

```
\ { { s where s is Element of RAT+ : s < t }
    where t is Element of RAT+ : t <> 0 };
```

Finally, real numbers are created by duplicating nonnegative real numbers to create non-positive numbers and then removing the non-positive zero so that zero has a unique representation. In the definition below, `[:{0},REAL+:]` should be understood as all the pairs $(0, x)$ with x a nonnegative real, while `[0,0]` is the pair $(0, 0)$.

```
func REAL -> set equals (REAL+ \/ [:{0},REAL+:]) \ {[0,0]};
```

Since negative integers and negative rational numbers were built the same way (a pair with a first element equal to zero / the empty set), integers and rational numbers are actual subsets of real numbers.

Arithmetic operations and their properties are first proved on Dedekind cuts, then extended to nonnegative reals, and finally extended to reals. For instance, the function below defines the addition on Dedekind cuts.

```
func A + B -> Element of DEDEKIND_CUTS equals
  { (r + s) where r, s is Element of RAT+ : ( r in A & s in B ) };
```

Finally, the library proves the completeness property under various forms. Below is the one for the least upper bound expressed as a function. Given a nonempty upper-bounded set, it returns the least upper bound (noted by the keyword `it` in the definition).

```
let X be real-membered set;
assume A1: ( not X is empty & X is bounded_above );
func upper_bound X -> real number means
  ( ( for r being real number st r in X holds r <= it ) &
    ( for s being real number st 0 < s holds
      ex r being real number st ( r in X & it - s < r ) ) );
```

3.2.6. *HOL4* As in Mizar's library, HOL4 defines real numbers (Harrison 1994) using Dedekind cuts:

```
val isacut = new_definition("isacut",
--`isacut C =
 (?x. C x)                              /\      (* Nonempty            *)
 (?x. ~C x)                             /\      (* Bounded above       *)
 (!x y. C x /\ y hrat_lt x ==> C y)     /\      (* Downward closed     *)
 (!x. C x ==> ?y. C y /\ x hrat_lt y)`--);  (* No greatest element *)
```

with `hrat_lt` the strict ordering on positive rational numbers. In this definition, a cut `C` is a subset of $\mathbb{Q}^+ \equiv \{(x+1, y+1) \mid x, y \in \mathbb{N}\}$.

To define operations and prove theorems, there are two functions `hreal`, from subsets of $\mathbb{Q}^+$ toward $\mathbb{R}^+$, and `cut` from $\mathbb{R}^+$ toward Dedekind cuts, characterized by

```
val hreal_tybij =
 define_new_type_bijections
   {name="hreal_tybij",ABS="hreal",REP="cut",tyax=hreal_tydef};
```

where `define_new_type_bijections` generates the functions `hreal` and `cut` such that

$$(\forall a,\ \texttt{hreal}(\texttt{cut}(a)) = a) \wedge (\forall r,\ \texttt{isacut}(r) \Leftrightarrow (\texttt{cut}(\texttt{hreal}(r)) = r))$$

Except for the multiplicative inverse, operations on positive real numbers are then defined in a way similar to Mizar:

$$\begin{aligned} \sup S &= \bigcup S \\ X + Y &= \{x + y \mid x \in X \wedge y \in Y\} \\ XY &= \{xy \mid x \in X \wedge y \in Y\} \\ X^{-1} &= \{w \mid \exists d < 1,\ \forall x \in X,\ wx < d\} \end{aligned}$$

Finally, as in HOL Light, real numbers are defined as the quotient of the set of pairs of positive real numbers by the following equivalence relation:

$$\forall x_1, y_1, x_2, y_2,\ \ ((x_1, y_1) \approx (x_2, y_2)) \Leftrightarrow (x_1 + y_2 = x_2 + y_1)$$

3.2.7. *ProofPower-HOL* As Mizar and HOL4, ProofPower-HOL defines real numbers using Dedekind cuts. The definition below uses ProofPower's syntax: • is the separator between quantifiers and formulas, \$≪ is an arbitrary relation (over which the formula is universally-quantified), and ≪ is the infix notation for \$≪.

$$\begin{aligned} \forall X\ \${\ll}\ A \bullet A \in \mathit{Cuts}(X, \${\ll}) \quad \Leftrightarrow \quad & A \subseteq X \wedge \neg A = \{\} \wedge \mathit{UnboundedAbove}(A, \${\ll}) \\ & \wedge (\exists x \bullet x \in X \wedge \mathit{UpperBound}(A, \${\ll}, x) \\ & \wedge (\forall a\ b \bullet a \in A \wedge b \in X \wedge b \ll a \Rightarrow b \in A)). \end{aligned}$$

Unlike those systems, here Dedekind cuts are subsets of dyadic numbers $\mathbb{D} = \{(2m + 1)/2^n \mid m \in \mathbb{N} \text{ and } n \in \mathbb{Z}\}$. The reference manual (ProofPower development team 2006) only describes multiplications and natural exponentiation for this set in theory `DYADIC`.

Unfortunately, the construction of negative real numbers and real operations is not described. Moreover, another formalization of real numbers, based on Dedekind cuts of $\mathbb{Z}\left[\sqrt{2}\right]$, is described by Arthan (2001), but is not available.

3.3. *Non-Standard Analysis*

Non-standard analysis introduces the notions of *standard* and *non-standard* functions and predicates. The terms *internal* or *classical* can also be encountered in place of standard. These notions are not related to classical logic: standard means that the formulas use features from standard analysis only. In particular, standard formulas neither test whether a number is standard nor do they extract its standard part. The main theorem from non-standard analysis is the *transfer* principle: any standard predicate with only standard constants that holds for all standard numbers holds for all numbers.

Now that the vocabulary is set, let us consider the systems that provide non-standard analysis. It is available in ACL2(r) and Isabelle/HOL.

3.3.1. *ACL2(r)* Since ACL2 makes it difficult to manipulate formulas with several quantifiers, it is not suitable to reason about real analysis, due to the usual formulas about

limits "$\forall \varepsilon > 0,\ \exists \delta > 0 \ldots$". To circumvent this limitation, rather than formalizing standard analysis, ACL2 has been extended so as to support non-standard analysis and thus avoid these quantifiers (Gamboa and Kaufmann 2001).

The modified system ACL2(r) defines three new symbols: predicate `standard-numberp` tests whether a number is standard, function `standard-part` returns the standard part of a number, and constant `i-large-standard` is a non-standard integer (hence larger than any standard integer). Any formula built on these symbols is non-standard.

Since it reuses the library of ACL2, all the axioms and theorems, which were implicitly quantified on rational numbers, now apply to hyper-reals.

```
(defaxiom Associativity-of-+
  (equal (+ (+ x y) z) (+ x (+ y z))))
```

The induction principle on natural numbers has to be modified though. In presence of a non-standard formula, it requires that the predicate be true for any non-standard integers (while, for standard ones, the traditional induction hypothesis $P(n) \Rightarrow P(n+1)$ applies). This keeps the system consistent but it also means that the induction principle is no longer usable for any formula that contains irrational numbers. Indeed, such numbers are non-standard from a syntactical point of view. For instance, Napier's constant e may be defined as the standard part of $\sum_{n=0}^{N} 1/n!$ with N equal to `i-large-standard`.

To avoid this issue, ACL2(r) provides an alternate command for defining functions (event `defun-std` instead of event `defun`), so that one can tell the system that a formula is actually standard even if it contains non-standard symbols. More precisely, given a non-standard function that returns standard numbers for any standard inputs, the axioms of non-standard analysis ensure that there exists a unique standard function that is equal to the given function on standard inputs. This standard function is the one created by `defun-std`, once the user has proved that outputs are standard whenever inputs are (Gamboa et al. 2004). In the case of the Napier's constant above, the function being a standard part, it trivially satisfies these hypotheses.

Note that the resulting function is usually not equal to the original function on non-standard inputs. For instance, the following function is provably the identity function (by the transfer principle) while `standard-part` is definitely not (Gamboa and Kaufmann 2001).

```
(defun-std std-pt (x) (standard-part x))
```

Similarly, the system proposes a new command for defining theorems (`defthm-std` instead of `defthm`) based on the transfer principle: a theorem holds if its statement is standard and the user proves that it holds for any standard inputs.

3.3.2. *Isabelle/HOL* The Isabelle/HOL library 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 in Isabelle/HOL (Fleuriot 2000). While originally about real numbers only, the definition has later been generalized to any type. The equivalence relation is defined as follows, with `\<U>` a free ultrafilter on the natural numbers (proved to exist by Zorn's lemma).

```
definition starrel :: "((nat => 'a) × (nat => 'a)) set"
  where "starrel = {(X,Y). {n. X n = Y n} ∈ \<U>}"
```

The resulting quotient type `'a star` is then specialized for real numbers:

```
type_synonym hypreal = "real star"
```

All the arithmetic and comparison operators are then lifted pointwise from reals to hyper-reals. Standard reals are defined as those that are equivalent to constant sequences:

```
definition star_of :: "'a => 'a star"
  where "star_of x == star_n (λn. x)"
definition Standard :: "'a star set"
  where "Standard = range star_of"
```

From the point of view of Isabelle/HOL, the transfer principle is neither an axiom nor a theorem, but a meta-theorem, since it applies to theorem statements. As such, it is not directly proved in Isabelle/HOL. Instead, a tactic transferring standard statements is defined; when it is applied, each of its steps is proved. This is standard LCF practice.

Note that, while later ported to the Isabelle/HOL reals based on Cauchy sequences (Section 3.2.2), this development was originally built on top of real numbers defined as Dedekind cuts. These cuts were defined in a way similar to HOL4's ones.

4. Formalizations of Real Analysis

We now study how the systems define and prove some usual notions of real analysis: sequences, series, continuity, differentiability, and integrals. In most formalizations, such notions are stated only in the specific case of functions from real or natural numbers to real numbers. HOL4, Isabelle/HOL, HOL Light, and C-CoRN define them in a more general context, such as metric or topological spaces. ACL2(r), as explained above, uses notions from non-standard analysis.

The impact of the underlying logic framework is mostly noticeable when formalizing real numbers. At the level of real analysis, the differences between provers subside and the design choices are mainly guided by the kind of analysis: constructive, standard, non-standard. So the section is subdivided between notions of real analysis instead.

4.1. *Sequences and Series*

Almost all the formalizations define real sequences as functions from natural numbers to real numbers. Only Mizar has a slightly different definition, due to its set theoretical approach to functions. Convergence definitions, however, differ.

4.1.1. *Real sequences* In the Coq standard library (Mayero 2001), C-CoRN/MathClasses, Mizar (Kotowicz 1990*a*), PVS (Dutertre 1996), and ProofPower (Arthan 2001), convergence of a real sequence (u_n) toward $\ell \in \mathbb{R}$ is defined with the usual property:

$$\forall \varepsilon \in \mathbb{R}^+,\ \exists N \in \mathbb{N},\ \forall n \in \mathbb{N},\ n \geq N \Rightarrow |u_n - \ell| < \varepsilon \tag{1}$$

where $\mathbb{R}^+$ is the set of positive reals. Note that the definition for C-CoRN/MathClasses works in any ordered field and not just real numbers; also it names this property `Cauchy_prop` while it is not equivalent to Cauchy property in non-complete spaces.

Definition (1) is dedicated to real sequences. But some properties can be formalized in a weaker scope, and thus be used later for convergence of real functions. For instance, in Isabelle/HOL, convergence of sequences is defined using convergence in topological spaces (Hölzl et al. 2013):

$$\forall V \in \{ \text{ opens } \},\ \ell \in V \Rightarrow u^{-1}(V) \in \mathcal{F} \tag{2}$$

where $\mathcal{F}$ is a proper filter, *i.e.*

$$\begin{aligned}
&\mathcal{F} \neq \emptyset,\ \emptyset \notin \mathcal{F},\\
&\forall A, B,\ A \in \mathcal{F} \wedge A \subset B \Rightarrow B \in \mathcal{F}, \text{ and}\\
&\forall A, B \in \mathcal{F},\ A \cap B \in \mathcal{F}.
\end{aligned}$$

For sequences, the chosen filter is $\mathcal{F} = \{A \subset \mathbb{N} \mid \exists N \in \mathbb{N},\ \forall n \in \mathbb{N},\ n \geq N \Rightarrow n \in A\}$. Definition (2) will also be used for convergence of real functions, by choosing a different filter (Section 4.2.2).

In HOL4 and HOL Light, the definition of convergence in a topological space is formalized using nets:

$$\forall N \in \{\text{neighborhood of } \ell\},\ \exists n,\ \forall m,\ m \succeq n \Rightarrow u_m \in N \tag{3}$$

where $\succeq$ is a partial order. For sequences, N is a neighborhood for the topology generated by the metric space $(\mathbb{R}, (x, y) \mapsto |y - x|)$ and $\succeq$ is the order on natural numbers. Moreover, function `lim` gives the limit if it exists using Hilbert's epsilon.

Finally, in ACL2(r), a sequence (u_n) converges to $\ell \in \mathbb{R}$ if $u_N \approx \ell$ (*i.e.* u_N and ℓ are infinitesimally close) for all non-standard natural numbers N (Gamboa and Kaufmann 2001). The limit is the standard part of u_N for a specific N. These definitions of convergence and limit show the point of choosing non-standard analysis for ACL2(r). Indeed, they make it possible to avoid quantifiers almost entirely.

For all these formalizations, compatibility of limit with arithmetic operations ($+$, $-$, $\times$, and $/$) and relations ($=$, $\leq$) is proved. They also provide the usual convergence theorems, *e.g.* the convergence of Cauchy sequences. While the notion of sub-sequences is useful to prove non-convergence of a sequence in practice, only C-CoRN and Mizar (Kotowicz 1990*b*) include it.

4.1.2. *Real series* Power series are the usual tool for defining transcendental functions such as exponential and trigonometric functions. So it does not come as a surprise that most formalizations provide an implementation of partial sums, series, and power series. Only Mizar, HOL Light, and C-CoRN/MathClasses, provide more theorems than what is strictly needed for defining elementary functions. PVS provides a development of about 300 lemmas about series and power series too, but they are not used in the main development about real analysis. For instance, there are two definitions of the real exponential function, which are not proved to be equivalent.

Series are defined, with few variations, with the usual definition:

$$\sum_{n\in\mathbb{N}} u_n \text{ is convergent if } n \mapsto \sum_{k=0}^{n} u_k \text{ is convergent.} \tag{4}$$

The Coq standard library and C-CoRN/MathClasses first define the partial sums starting from 0 and use them to define other partial sums. In the standard library, the partial sum $\sum_{k=n}^{m} u_k$ is defined as $\sum_{k=0}^{m-n} u_{k+n}$. Notice that, when $m < n$, the natural difference $m - n$ is 0, so the sum between n and m is u_n, rather than 0 or the opposite of another sum. Convergent series use an ad-hoc predicate rather than being defined as the convergence of the sequence of partial sums. This unfortunate choice prevents the reuse of theorems on sequences.

In C-CoRN, $\sum_{k=n}^{m} u_k = \sum_{k=0}^{(m+1)-1} u_k - \sum_{k=0}^{n-1} u_k$ where (u_n) takes value in an abelian group. This time, the definition for the case $m < n$ is a bit more sensible. The convergence of series is given by Definition (4).

In HOL4 (Harrison 1998), Isabelle/HOL, and PVS (Gottliebsen 2000), which formalizations are based on Harrison's,

$$sum(u, n, m) = \begin{cases} 0 & \text{if } m = 0 \\ u_{n+m-1} + sum(u, n, m-1) & \text{else} \end{cases}$$

With this definition, having a partial sum $\sum_{k=n}^{m} u_k$ with $m < n$ is no longer possible, but the way bounds are handled is a bit less natural, since the partial sum is now $sum(u, n, m - n + 1)$. The convergence of partial sums is the convergence of sequences described in Section 4.1.1. In these formalizations, the d'Alembert criterion and convergence of alternated series are proved. In HOL4 and Isabelle/HOL, a function that returns the limit of partial sums is defined using Hilbert's epsilon.

In HOL Light, sums have been generalized to having indexes in arbitrary finite sets and there are combinatorial theorems to manipulate them, thus avoiding the above inconvenience.

Mizar (Raczkowski and Nedzusiak 1991*b*) and ProofPower (Arthan 2012) define series using explicitly Definition (4). Various convergence theorems are proved, from the d'Alembert criterion to comparison theorems.

The Cauchy product is provided by these formalizations as it is used to prove the usual property on exponential $\forall x, y \in \mathbb{R},\ e^{x+y} = e^x \cdot e^y$:

$$\left(\sum_{n\in\mathbb{N}} a_n\right)\left(\sum_{n\in\mathbb{N}} b_n\right) = \sum_{n\in\mathbb{N}}\left(\sum_{k=0}^{n} a_k b_{n-k}\right). \tag{5}$$

ACL2(r) does not formalize partial sums and series (Gamboa and Kaufmann 2001); it uses only sequences to define exponential and trigonometric functions.

4.1.3. *Power series and series of functions* Power series are defined as functions $f : x \mapsto \sum_{n\in\mathbb{N}} a_n x^n$ where the sequence (a_n) takes value in real numbers or in an abelian group. These functions are defined for x such that $\sum_{n\in\mathbb{N}} a_n x^n$ is convergent.

HOL Light provides an advanced formalization of complex analysis, *e.g.* equivalence

between holomorphic and analytic functions. As such, this prover comes with a fully-featured library when it comes to power series and series of functions.

Other systems are much less comprehensive. Indeed, in order to define exponential and trigonometric functions, one does not need many theorems about sequences and series of functions. As a consequence, most provers provide just the definition of power series and some theorems about them. This is the case for HOL4 (Harrison 1994) and PVS (Gottliebsen 2000). They provide some convergence criteria for power series, *e.g.* d'Alembert's. They also have a notion of convergence circle. HOL4 provides a theorem about the differentiability of power series.

Another way to represent power series is as series of functions, as is done in Coq, C-CoRN, Isabelle/HOL, Mizar (Perkowska 1992), and ProofPower (Arthan 2012) For instance, usual theorems about continuity and differentiability for limits of sequences and series are proved and used to study exponential and trigonometric functions. Coq provides a definition of power series and the d'Alembert criterion.

4.2. *Functions over Real Numbers*

We now survey the notions of analysis available to study functions of $\mathbb{R}$ to $\mathbb{R}$: partial functions, limits, continuity, differentiability.

4.2.1. *Partial functions* Mathematical functions are usually defined as functions from a subset of real numbers to real numbers. There are two ways for proof systems to define partial functions. They either specify the definition domain in the type of the function, or they use total functions and specify the domain in theorem hypotheses. For instance, the square root function can be defined, either as a partial function defined on nonnegative reals, or as a total function defined for all real numbers. In the latter case, a theorem such as $\sqrt{x}^2 = x$ will have $x \geq 0$ as an explicit hypothesis. The way of defining functions is highly dependent on the underlying logic.

Mizar identifies functions with their graphs: sets X such that $\forall x, y_1, y_2,\ \langle x, y_1\rangle \in X \wedge \langle x, y_2\rangle \in X \Rightarrow y_1 = y_2$. Thanks to this approach, the domain is defined as $\{x \mid \exists y,\ \langle x, y\rangle \in X\}$ and $f(x)$ is the only value such that $\langle x, f(x)\rangle \in X$ if x is in the domain (Byliński 1990). Due to the set-theoretical approach of Mizar, having functions defined on partial domains is natural.

PVS also uses partial domains: a real function is a function defined on a sub-type of real numbers. Using type correctness conditions (TCC), side conditions are automatically generated, and often proved for simple functions. This definition allows one to formalize theorems without making a distinction between total and partial functions. Again, this choice is natural for PVS, due to the pervasiveness of sub-types in its logic framework.

In C-CoRN, functions have to be compatible with the equivalence relation, so that a function applied to two different Cauchy sequences representing the same real numbers gives the same result. So each function has to come with this property of being well-defined. For partial functions, the only difference is that this property is only required for values in the domain, which also has to be well-defined.

HOL-based systems, including ProofPower-HOL, use the second approach: functions

are total and theorems work only on restricted domains. Indeed, sub-typing is hardly supported in these systems. So it is simpler to ensure that all real functions have the same type, that is, their preimage is $\mathbb{R}$. This is also the case for some functions of the Coq standard library such as the multiplicative inverse and the square root.

In ACL2, the way to define partial functions is on a case-by-case basis. The different approaches are described by Manolios and Moore (2003). The situation is sensibly different from other systems because the issue of function sub-typing is nonexistent: manipulating functions is outside the scope of a first-order system.

4.2.2. *Limits* The notion of limit is defined in PVS and ProofPower (Arthan 2012) using the usual ε-δ definition. In a domain D, a function $f : \mathbb{R} \to \mathbb{R}$ converges toward a limit $\ell \in \mathbb{R}$ at point $x \in \overline{D}$ when

$$\forall \varepsilon \in \mathbb{R}^+,\ \exists \delta \in \mathbb{R}^+,\ \forall y \in D,\ |y - x| < \delta \Rightarrow |f(y) - \ell| < \varepsilon. \tag{6}$$

In Coq's standard library, the convergence is given for arbitrary metric spaces. For a function $f : (X, d_X) \to (Y, d_Y)$, it is defined by Mayero (2001) as

$$\forall \varepsilon \in \mathbb{R}^+,\ \exists \delta \in \mathbb{R}^+,\ \forall y \in D,\ d_X(x, y) < \delta \Rightarrow d_Y(\ell, f(y)) < \varepsilon. \tag{7}$$

This definition leaves out the condition "x is a limit point of D". This makes the definition easier to use, but uniqueness of the limit ℓ is no longer guaranteed.

In Isabelle/HOL and HOL Light, convergence for functions between two topological spaces is defined using the topological definition (2) with the filter $\mathcal{F} = \{P \mid \exists S \in \{\text{opens}\}, x \in S \wedge S \backslash \{x\} \subseteq P\}$. The formalization proves that this definition is equivalent to (7) in a metric space.

HOL4 uses its topological definition (3) of convergence with the same topology as for convergence of sequences, but with the partial order defined by $\forall y, y' \in \mathbb{R},\ y \succeq y' \Leftrightarrow |y - x| \leq |y' - x|$. Unlike sequences and series, there is no notation for limits of functions.

The notion of convergence for real functions in Mizar (Kotowicz 1991*a*) is defined using sequences as follows:

$$\left\{ \begin{array}{l} x \in \overline{\text{dom } f} \\ \exists \ell \in \mathbb{R},\ \forall (x_n) \in \mathbb{R}^{\mathbb{N}}, \end{array} \begin{array}{c} \forall n \in \mathbb{N}, x_n \in \text{dom } f \backslash \{x\} \\ \wedge \quad \lim\limits_{n \to +\infty} x_n = x \end{array} \right\} \Rightarrow \lim_{n \to +\infty} f(x_n) = \ell. \tag{8}$$

The equivalence with (6) is proved in the particular case $D = \text{dom } f \backslash \{x\}$. Mizar also formalizes notions of limit at infinity (Kotowicz 1991*b*) and one-side limits (Kotowicz 1991*c*).

C-CoRN formalizes Definitions (6) and (8) for total functions and proves their equivalence. Limits for partial functions are not defined.

Finally, in ACL2(r) and in the non-standard analysis library for Isabelle/HOL, limits are defined using non-standard real numbers:

$$\forall y,\ y \approx x \Rightarrow f(y) \approx \ell. \tag{9}$$

4.2.3. *Continuity* The notion of continuity of function f at point x can be defined using the limit: $\lim_{y \to x} f(y) = f(x)$. This is the case for HOL4 and HOL Light.

Continuity of f at x can also be defined using an ad-hoc predicate:

$$\forall \varepsilon \in \mathbb{R}^+,\ \exists \delta \in \mathbb{R}^+,\ \forall y \in \mathbb{R},\ |y - x| < \delta \Rightarrow |f(y) - f(x)| < \varepsilon. \tag{10}$$

PVS adapts this definition to partial functions and proves the preservation of continuity by basic operations (Dutertre 1996).

In the Coq standard library, both variants are formalized and the equivalence is proved. Since the generic limit (7) is only defined for metric spaces and not vector spaces, continuity theorems (*e.g.* continuity of linear combination) have been proved only for the ad-hoc definition though. Definition (10) is the main one: it is used in most theorems that need some continuity hypothesis.

As for limits, Mizar uses sequences to define continuity (Raczkowski and Sadowski 1990*a*):

$$\forall (x_n) \in \mathbb{R}^{\mathbb{N}},\ (\forall n \in \mathbb{N}, x_n \in \text{dom } f) \wedge \lim_{n \to +\infty} x_n = x \Rightarrow f(x) = \lim_{n \to +\infty} f(x_n). \tag{11}$$

This definition is proved equivalent with (10) and with the topological version (2). ProofPower (Arthan 2012) goes the other way around: it uses Definition (10) and proves the equivalence with the sequence-based version of continuity (11).

The case of C-CoRN is peculiar, since it formalizes constructive mathematics. As such, any function is naturally continuous. (Otherwise, given a discontinuous function, one could devise a way to decide equality between real numbers.) Therefore, having a predicate for continuity is useless; it is replaced by uniform continuity, which is formalized for partial functions on a closed interval $[a; b] \subseteq \text{dom } f$.

4.2.4. *Differentiability* For differentiability, Coq's standard library, PVS, and ProofPower (Arthan 2012), use an ε-δ formula based on Newton's difference quotient:

$$\forall \varepsilon \in \mathbb{R}^+,\ \exists \delta \in \mathbb{R}^+,\ \forall y \in D \backslash \{x\},\ |y - x| < \delta \Rightarrow \left| \frac{f(y) - f(x)}{y - x} - \ell \right| < \varepsilon \tag{12}$$

directly or through the limit definition.

Note that, in Coq, there is no simple way to express the value $\ell = f'(x)$ above. Indeed, the term `derive_pt` f x is not a real number but a function that takes a proof that f is derivable at x and returns a real number. As a consequence, whenever one wants to write $f'(x)$ in a formula, one has to explicitly build a proof of differentiability beforehand and pass it (Boldo et al. 2012).

In HOL4 and HOL Light, Newton's difference quotient is used too, but through their respective topological limit definitions ((2) and (3)). Another approach is implemented in additional libraries of HOL Light: the Frechet derivative in a real normed vector space (Harrison 2005; 2013). In C-CoRN, as with continuity, the previous definition is modified to get a uniform differentiability on a closed interval $[a; b]$, *i.e.* there is a single δ for all points of $[a; b]$ (Cruz-Filipe 2004). In Mizar, differentiability is defined for multivariate functions as usual by the existence of a linear function L such as $f(x+h) - f(x) - L(h) = o(h)$ (Raczkowski and Sadowski 1990*b*).

In the non-standard analysis libraries of Isabelle/HOL (Fleuriot 2000) and ACL2(r)

(Gamboa 2000), differentiability is defined by applying Definition (9) of limit to Newton's difference quotient.

In all these formalizations, usual theorems such as the Mean Value Theorem and the Intermediate Value Theorem are proved.

4.3. *Integrals*

All the definitions of limit seen previously are equivalent in the scope of real numbers; this is also true for differentiability. For integrals, however, the integrability of some functions depends on the actual definition. The more usual definitions in mathematics are the Riemann integral and the Lebesgue integral. Except for ACL2(r), all libraries formalize at least one of these integrals. Only PVS defines both: the Riemann integral is the most used for real analysis, while the Lebesgue integral is preferred for probability theory; PVS proves the equivalence between these integrals whenever they are both defined.

All the integrals presented below are formalized for the preferred type of functions of each provers: total functions for Coq's standard library and HOL-based provers, and partial functions for Mizar, PVS, and C-CoRN/MathClasses.

4.3.1. *The Riemann integral* To define the Riemann integrability in Coq, Desmettre (2002) defines the integral on step functions and then uses the traditional ε-δ definition:

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

The value of the integral is then defined as the limit of $\int \varphi$ when $\varepsilon \to 0$. As with derivatives, the Riemann integral in Coq needs a proof that the function is integrable.

PVS (Butler 2009) and Mizar (Endou and Korniłowicz 1999) define integrability as the convergence of Riemann sums $S(f, \sigma, \xi) = \sum_{i=0}^{n} (\sigma_{i+1} - \sigma_i) f(\xi_i)$ where σ and ξ are finite sequences such that $\forall i \in [\![0, n]\!],\ \sigma_i \leq \xi_i \leq \sigma_{i+1}$. Both definitions are mathematically equivalent. Coq's formula, however, cannot be used to effectively compute the value of the integral.

C-CoRN defines the Riemann integral between a and b, $a \leq b$, for uniformly continuous functions as the limit of the sequence of Riemann sums where $\sigma = (a + k \cdot \frac{b-a}{n+1})_{0 \leq k \leq n+1}$ and $\xi = (a + k \cdot \frac{b-a}{n+1})_{0 \leq k \leq n}$. As with convergence, there is a problem with naming: C-CoRN names these sums "Darboux sums" instead of "Riemann sums". The library provides no notion of integrability because C-CoRN defines the integral only for uniformly continuous functions.

MathClasses defines the Riemann integral too. It even defines the Stieltjes integral (O'Connor and Spitters 2010). 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.

4.3.2. *The Henstock-Kurzweil integral* HOL4 (Shi et al. 2013), HOL Light (Harrison 2013), Isabelle/HOL, and ProofPower (Arthan 2012), provide a definition of the Henstock-

Kurzweil integral, or Gauge integral. It is a generalization of the Riemann integral: I_f is the Henstock-Kurzweil integral of a function f in the interval $[a;b]$ if

$$\begin{aligned}\forall \varepsilon \in \mathbb{R}^+, \exists g : [a;b] \to \mathbb{R}^+, \forall \sigma \in \{\text{subdivision of } [a;b]\},\ \forall \xi \in \mathbb{R}^{|\sigma|-1},\\ (\forall i < |\sigma|,\ \sigma_i \leq \xi_i \leq \sigma_{i+1} \wedge \sigma_{i+1} - \sigma_i < g(\xi_i)) \Rightarrow |S(f,\sigma,\xi) - I_f| < \varepsilon \end{aligned} \quad (13)$$

If we restrict functions g to constant functions, this is equivalent to the Riemann integral defined as the limit of Riemann sums.

4.3.3. *Measure theory and the Lebesgue integral* HOL4 (Mhamdi et al. 2010), Isabelle/HOL (Hölzl and Heller 2011), Mizar, and PVS, provide a library about measure theory and use it to define integrals and probability theory. These two formalizations first define a σ-algebra $\mathcal{A}$ of a set X as follows:

$$\begin{cases} \emptyset \in \mathcal{A} \\ \forall A \in \mathcal{A},\ X \backslash A \in \mathcal{A} \\ \forall (A_n) \in \mathcal{A}^{\mathbb{N}},\ \bigcup_{n\in\mathbb{N}} A_n \in \mathcal{A} \end{cases} \quad (14)$$

A measure $\mu : (X, \mathcal{A}) \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$\begin{cases} \mu(\emptyset) = 0 \\ \forall\, (A_n) \in \mathcal{A}^{\mathbb{N}},\ (\forall i,j \in \mathbb{N},\ i \neq j \Rightarrow A_i \cap A_j = \emptyset) \Rightarrow \displaystyle\sum_{n\in\mathbb{N}} \mu\,(A_n) = \mu\left(\bigcup_{n\in\mathbb{N}} A_n\right) \end{cases} \quad (15)$$

These definitions require the use of extended real numbers. HOL4 and Isabelle/HOL provides $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty, -\infty\}$ as an algebraic datatype. Order, addition, and multiplication, are defined on it. PVS only provides the notion of nonnegative extended reals as pairs of a Boolean and a real number, since it is sufficient for defining a measure. Moreover, addition and multiplication are easier to define for $\mathbb{R} \cup \{+\infty\}$ than for $\overline{\mathbb{R}}$.

In these formalizations, integrals are defined for simple functions as

$$\int \left(\sum_{k=0}^{n} \alpha_i \chi_{A_i}\right) d\mu = \sum_{k=0}^{n} \alpha_i \mu(A_i) \quad (16)$$

and for nonnegative functions as

$$\int f\, d\mu = \sup\left\{ \int g\, d\mu \;\middle|\; g \text{ is a simple function and } g \leq f \text{ except on a null set} \right\}. \quad (17)$$

Finally, a function f is integrable if the integrals of both $f^+ = \max\{f, 0\}$ and $f^- = \max\{-f, 0\}$ are finite. Its value is then

$$\int f\, d\mu = \int f^+\, d\mu - \int f^-\, d\mu. \quad (18)$$

In all these formalizations, the Lebesgue integral is defined using these integrals on the σ-algebra generated by intervals and using the Lebesgue measure generated by $\lambda\,([a;b]) = b - a$. Measure theory is also used to formalized probabilities in these systems.

The approach of HOL Light is slightly different. The measure of a set is defined as the Gauge integral of the constant function 1 over this set. Moreover, a function is

measurable if there is a sequence of continuous functions that converges toward it except on a negligible set (Harrison 2013).

ACL2(r) provides a formalization of Lebesgue measure, but only proves that there exists some set of real numbers without Lebesgue measure (Cowles and Gamboa 2010). There is no integral in this system.

4.3.4. *Fundamental Theorem of Calculus* All the systems that define an integral provide a proof, under various conditions, of one of these relations between derivative and integral:

$$\int_a^b f'(t)\,dt = f(b) - f(a) \tag{19}$$

$$\text{or} \qquad \frac{d}{dx}\left(x \mapsto \int_a^x f(t)\,dt\right) = f(x) \tag{20}$$

Formula (19) is the Fundamental Theorem of Calculus. This theorem is proved in all the surveyed formalizations for their main integral (Harrison 1998, Shi et al. 2013, Endou et al. 2001, Cruz-Filipe 2003, Butler 2009, Arthan 2012, Kaufmann et al. 2000) except Coq's standard library. Except for systems using Gauge integrals, the theorem requires that f be continuous on the interval $[a;b]$.

In Coq's standard library, due to difficulties to work with partial functions, (20) requires that f be continuous on $[a;b]$ with $b > x$, rather than just continuous at point x and integrable around it. Note that this library also provides the definition of the Newton integral (*i.e.* a function g is *Newton integrable* if there is a function f differentiable such that $f' = g$). For this particular integral, (19) is the definition of the integral of g.

4.4. *Elementary Functions*

Finally, the notions formalized above are used to define some elementary functions: exponential, logarithm, trigonometric functions, and their inverses. Most provers provide all these functions but their definitions present some variations, *e.g.* by power series or integrals.

4.4.1. *Using power series* As written above, series and power series are generally defined to formalize exponential, sine, and cosine. Coq's standard library (Mayero 2001), C-CoRN, HOL4 (Harrison 1994), the sub-library about series of PVS (Owre and Shankar 2003, Gottliebsen 2000), the PVS constructive reals, and ACL2(r) (Gamboa and Kaufmann 2001), use the following power series to define them:

$$\exp(x) = \sum_{n\in\mathbb{N}} \frac{x^n}{n!}, \qquad \sin(x) = \sum_{n\in\mathbb{N}} \frac{x^{2n+1}}{(2n+1)!}, \qquad \cos(x) = \sum_{n\in\mathbb{N}} \frac{x^{2n}}{(2n)!}.$$

In MathClasses, exp, sin, and arctan, are defined using power series (O'Connor 2008, Krebbers and Spitters 2013) and the other trigonometric functions, such as cos or ln are defined from these three. Then π is defined from arctan using a Machin-like formula, as in PVS' constructive reals.

HOL Light defines the exponential function with power series over complex numbers

and then defines the exponential on real numbers and the trigonometric functions from it. Mizar uses the same approach for defining the trigonometric functions, but not for the exponential on real numbers (*cf.* Section 4.4.4). These two formalizations are the only ones to define power series on complex numbers.

4.4.2. *Using integrals* In PVS and C-CoRN, natural logarithm is defined for any positive number x using the Riemann integral:

$$\ln x = \int_1^x \frac{dt}{t}.$$

Moreover, PVS and Mizar also define the inverse of tangent using integrals:

$$\arctan(x) = \int_0^x \frac{dt}{1+t^2}.$$

This definition of arctan is used to define inverse functions of sine and cosine.

4.4.3. *Using inverse functions* ACL2(r) defines a general inverse (Gamboa and Cowles 2009) for functions that are both surjective and injective. As most usual functions are continuous, it is natural to use the Intermediate Value Theorem to prove surjectivity and then define an inverse function for continuous functions. ACL2(r) uses this definition to formalize natural logarithm and inverses of trigonometric functions.

HOL Light, Isabelle/HOL, Coq, Mizar (Raczkowski and Nedzusiak 1991*a*), and ProofPower (Arthan 2012), also define natural logarithm (and general logarithm for Mizar) using the inverse of exponential function but without defining a general theory for inverse functions.

Contrarily, PVS defines exponential as the inverse function of natural logarithm. It also defines sine and cosine using inverse functions of arcsin and arccos.

4.4.4. *Other approaches* Mizar (Raczkowski 1991) defines general exponential functions using limits:

$$\forall a, b \in \mathbb{R},\ \forall s \in \mathbb{Q}^{\mathbb{N}},\ \lim_{n\to+\infty} s_n = b \Rightarrow \lim_{n\to+\infty} a^{s_n} = a^b \tag{21}$$

where $\forall a \in \mathbb{R},\ \forall (p,q) \in \mathbb{Z} \times (\mathbb{N}\backslash\{0\}),\ a^{\frac{p}{q}} = \sqrt[q]{a^p}$. Napler's constant e is defined as the limit of the convergent sequence $n \mapsto (1+\frac{1}{n+1})^{n+1}$.

To define sine and cosine, Mizar and HOL Light use the complex exponential function, defined by power series, with Euler formulas:

$$\sin(x) = \frac{e^{ix} - e^{-ix}}{2}, \qquad \cos(x) = \frac{e^{ix} + e^{-ix}}{2}.$$

In all studied formalizations, the tangent function is defined as the usual quotient $\tan(x) = \frac{\sin(x)}{\cos(x)}$.

ProofPower (Arthan 2012) defines the exponential function using differential equations: it is the solution of $f' = f$ such that $f(0) = 1$. Functions sine and cosine are defined as solution of a system of differential equations: $\sin(0) = 0,\ \cos(0) = 1,\ \sin' = \cos$ and

$\cos' = -\sin$. This system uses an unusual way to define π: it is the positive generator of the group of roots of the sine function, *i.e.* $\pi > 0 \wedge \pi\mathbb{Z} = \{x \in \mathbb{R} \mid \sin x = 0\}$.

Note that older versions of PVS were using an axiomatization for trigonometric functions. For instance, they were postulated as satisfying the identities $\sin^2(a) + \cos^2(a) = 1$ and $\cos(a) = \sin(a + \pi/2)$. Remnants of this formalization are still provided by recent versions.

5. Methods of Automation

For the sake of usability, formal systems come with various methods designed to reduce the amount of proofs or intermediate lemmas a user needs to write to formally verify a theorem. For instance, systems implementing a declarative style of proofs, *e.g.* Mizar, will try to automatically instantiate sequences of theorems so as to find the deduction steps for going from a user assertion to the next one. Since it does not provide a proof script language, ACL2 provides the same kind of automated instantiations. In all of these provers, developers of formalizations may have to prepare databases of facts for the systems to choose from.

Some systems also provide decision procedures, possibly incomplete and nonterminating, that a user can apply to complete a proof branch. For instance, they may perform propositional and first-order resolution, BDD-based simplifications, Prolog-like proof search, normalization according to rewrite systems, SMT solving, and so on. We will not describe any of these generic procedures, but instead focus on the ones that have been purposely designed to tackle real arithmetic and analysis.

Note that various words are used to describe such automatic methods depending on the system: strategy, tactic, method, and so on. We will use them interchangeably.

5.1. *Real Arithmetic*

First, let us have a look at methods dedicated to proving properties on rational and real numbers. This section is not meant to be exhaustive, especially as systems like HOL Light and Isabelle/HOL are generous when it comes to decision procedures. Instead, it is meant to give an overview about the capabilities of various systems.

Note that systems may also provide some help in manipulating expressions. For instance, in PVS, the Manip arithmetic package (Vito 2003) does not aim at automatically solving goals but at an easy handling of both equalities and inequalities. This set of algebraic manipulation strategies allows one, for example, to easily move terms from one side to the other or to cancel terms. There is no heavy automation here, but a practical handling of equalities and inequalities that comes close to pen-and-paper manipulations.

5.1.1. *Algebraic equalities* Presumably, the feature the most sought of when it comes to real numbers is the ability to perform algebraic rearrangements to prove universal equalities over real numbers.

The Coq proof assistant comes with a tactic `ring` that makes it possible to automatically prove equalities between two polynomials, *e.g.* $a^2 - b^2 = (a + b)(a - b)$. This tactic

behaves by first normalizing each side of the equality and then comparing them (Grégoire and Mahboubi 2005). The `field` variant of the tactic can cope with divisions; it normalizes the goal to rational functions instead of polynomials (Delahaye and Mayero 2001). The `ring` tactic is quite generic and has been instantiated on the real numbers from the MathClasses library.

PVS provides a `FIELD` strategy similar to Coq's `field`, but adapted to PVS specificities (Muñoz and Mayero 2001). The basic `GRIND` strategy is extended with additional theories `reals_props` and `extra_reals_props` as `GRIND-REALS`. It is very efficient, but as `GRIND`, it may not terminate.

The `REAL_RING` and `REAL_FIELD` decision procedures of HOL Light, the `algebra` strategy of Isabelle/HOL, and the `nsatz` tactic of Coq play a similar role. They are slightly more powerful though, as they do not normalize terms in isolation. Instead they use Gröbner bases and thus can handle the equational theories of integral domains and fields (Chaieb and Wenzel 2007, Pottier 2008).

5.1.2. *Linear inequalities* Taking into account the fact that the field of real numbers is ordered requires more complicated procedures. So, before tackling polynomial inequalities, we first consider only procedures dedicated to linear inequalities, *e.g.* $2x + 3y < 3z \Rightarrow y \geq z \Rightarrow 5x < 0$.

Coq provides the `fourier` tactic (superseded by `lra` in recent versions) based on Fourier-Motzkin quantifier elimination for solving universally-quantified systems of linear inequalities. HOL Light provides a similar algorithm with `REAL_ARITH`. This procedure is also able to perform some algebraic manipulations and to handle functions like maximum and absolute value though.

Isabelle/HOL provides a `ferrack` tactic that provides a quantifier elimination inspired by Ferrante and Rackoff's algorithm (Chaieb 2008). While it only tackles linear inequalities, the actual coefficients can be polynomials. Another provided procedure is `mir`; it is designed to solve linear systems that combine both real arithmetic and a floor function (Chaieb 2006).

For ACL2(r), the decision procedures of ACL2 for linear arithmetic over rational numbers are available (Hunt et al. 2003). As for PVS, the `SIMPLIFY` command has a decision procedure for linear inequalities, and thus, so do commands built upon it, *e.g.* `GRIND`.

5.1.3. *Other inequalities* Finally, let us have a look at polynomial and transcendental inequalities.

HOL Light comes with a quantifier-elimination procedure for real arithmetic named `REAL_ELIM_CONV` (McLaughlin and Harrison 2005). Such procedures are known to have a high complexity (doubly exponential time), which may prevent their use in practice. In some cases, especially univariate, a decision procedure based on *Positivstellensatz* refutations and sums of squares may be more efficient to prove systems of polynomial inequalities. HOL Light was the first system to come with such a procedure (Harrison 2007); the `REAL_SOS` function has not been integrated to the core system though. This implementation was later ported to Coq (`psatz`, Besson (2006)) and Isabelle/HOL (`sos`). All of these decision procedures rely on `csdp`, an external solver for semi-definite program-

ming problems. Due to the approximated computations of this solver, these procedures might fail to deduce a correct sum of squares and thus to prove valid goals. There exist improvements to this approach but they have not yet made their way into the procedures above (Monniaux and Corbineau 2011).

PVS provides some strategies based on numerical computations: `numerical` performs interval arithmetic to verify inequalities involving transcendental functions (Daumas et al. 2009); `bernstein` performs global optimization based on Bernstein polynomials to verify systems of polynomial inequalities (Muñoz and Narkawicz 2013).

The computational power of C-CoRN allows one to prove polynomial and transcendental inequalities just by computing the values with enough precision as long as there are no indeterminates. This feature may even be used in a proof based on the standard library by using the equivalence lemma proved in Kaliszyk and O'Connor (2009). Such computations are also available for the other formalizations based on Cauchy sequences, though they may be less efficient

5.2. *Continuity and Differentiability*

When it comes to real analysis, one often needs to prove that some given function is continuous or differentiable. As these goals are plentiful yet usually straightforward to prove, many systems provide automatic methods for discharging them.

5.2.1. *PVS* An automatic strategy for checking continuity is described in Gottliebsen (2000): the idea is to syntactically take the term apart in terms of addition, subtraction, multiplication, absolute value, and division (checking the denominator is non-zero). This approach based on strategies has been superseded by the use of judgments, to help the `GRIND` command apply the lemmas. An initial strategy may be used to instantiate the parametrized theories correctly before calling `GRIND`. Limitations of this approach include composed functions and functions with a trigonometric function in the denominator.

5.2.2. *HOL Light* In HOL Light (Harrison 1998), `DIFF_CONV` proves results about the derivative of expressions. It syntactically and repeatedly applies the rules for differentiating sums, differences, products, and quotients. It also knows about the derivatives of basic functions like x^n and some elementary functions, and the user may augment this set of functions. The derivative may also be postulated by the system.

5.2.3. *Coq's standard library* The Coq standard library provides a similar tactic named `reg`. It simplifies the goal as much as possible using the elementary continuity and differentiability rules. This tactic deals with all the forms of continuity (one point, all points) and differentiability (all points, one point, with or without the value of the derivative). It lacks extensibility.

5.2.4. *C-CoRN* Several tactics are also available in the C-CoRN library. The first ones are `Contin` and `Deriv` and consist in considering the application of a list of lemmas (the elementary rules) according to the form of the goal. Advanced tactics were developed:

the `New_Contin` and `New_Deriv` tactics aim at solving goals that state continuity and differentiability. They abstract the goal and keep track of differentiability and continuity domains during computations.

5.2.5. *ACL2(r)* This system has a `defderivative` event (Reid and Gamboa 2011). Given a function, it defines a function equal to the derivative of the input. In practice, the value is obtained automatically and usually does not match user expectations, but the user may simply state the equivalence with the provided form, which is much easier to prove.

The implementation of this event is based on the elementary algebraic differentiation rule for identity, constant, addition, multiplication, composition, and inverse. There is also a set of registered functions with their derivative, such as elementary functions, and the user may add more. This event depends on a proof of the derivative of exp, which was introduced by Reid and Gamboa (2011). The domain of differentiability used to be an interval but it is now a domain function. That means it cannot be quantified over anymore, but according to the authors, this choice adds flexibility and eases automations. As for the correctness, this event is not proved correct, but it generates a correctness proof when asked for, which is checked by ACL2(r).

As the prover is only first-order and the differentiation rules are not, they had to be encapsulated as constraints to emulate their behavior. Moreover, because of the non-standard analysis setting, partial derivatives are supported only for functions of up to two variables.

6. Conclusion

Table 1 summarizes this survey: it gives an overview of the various characteristics of the provers and libraries. It shows that formal systems come with a variety of theories of real numbers that cover the whole spectrum of the usual approaches for defining them: axiomatizations, or constructions as Dedekind cuts or Cauchy sequences. Note that C-CoRN is the only formalization of constructive analysis among them. This variety is also due to the variety of logical frameworks: the logic setting has a heavy influence on the way real analysis can be formalized.

This variety of formalizations might cause issues for the users as some approaches are better suited for some properties and no system provides all of them. This will also be a source of concern for people interested in exchanging theorems and proofs between various formal systems, such as Hurd (2011) for the HOL family of theorem provers. Hopefully, this survey provides enough details to help in choosing a system good enough for proving properties related to real analysis. More details on some historical and logical aspects of formalizing real numbers are detailed by Mayero (2012). To see how various provers deal with real analysis, one can also take a look at Wiedijk's webpage[§§]; he presents 100 famous theorems (some of them from real analysis) and their formal proofs using several systems.

[§§] `http://www.cs.ru.nl/~freek/100/`

	ACL2(r)	Coq	C-CoRN	HOL4	PPHol	Isabelle	HOLL	Mizar	PVS
Logic	FOL	deptypes	deptypes + constr	HOL	HOL	HOL	HOL	ZF	TCC
Partial functions	n.a.	total func + deptypes	deptypes	..... total functions, Hilbert-ε				function graphs	TCC
Definition of reals	non-std analysis	axiomatic	Cauchy sequences	Dedekind cuts	Dedekind cuts	Cauchy + UF	Cauchy sequences	Dedekind cuts	axiomatic
Integrals	n.a.	Riemann	Riemann Stieltjes	Gauge Lebesgue	Gauge	Gauge Lebesgue	Gauge Lebesgue	Riemann Lebesgue	Riemann Lebesgue
Multivariate analysis	1D	1D	2D	1D	1D	nD+	nD	nD+	2D
Dedicated automation	+	++	+	++	+	++	++		++

Table 1. Formalizations of real analysis at a glance.

The "logic" row gives a simplified vision of the logical framework: "deptypes" stands for dependent types, "constr" for constructive, and "ZF" for the Zermelo-Fraenkel set theory. The "partial functions" row is closely related, as it explains how partial functions are handled, *e.g.* division or derivative. The "definition of reals" row summarizes how real numbers are defined: "Cauchy sequences" and "Dedekind cuts" mean that they are built accordingly; "axiomatic" means that they are given by axioms; "non-std analysis" means that the real numbers are built given axiomatized non-standard predicates; "UF" means that there are non-standard reals built using the ultrafilter construction. The "integrals" row tells which definitions of integral are available. The "multivariate analysis" row describes its availability: "1D" means one-dimension only; "2D" means some results have been extended to $\mathbb{R}^2$; "nD" means a proper multivariate analysis; "nD+" means that some results are available for topological spaces other than $\mathbb{R}^n$. The "dedicated automation" row tries to evaluate the number, power, and breadth of automatic methods for real numbers and real analysis: the more "+" the better.

One of the lessons learned is that real analysis in ACL2(r) is not as developed as in other systems. Moreover, the absence of higher-order logic makes it difficult to manipulate functions. As such, this system should be reserved to users who have to interface with other ACL2 developments. Let us now examine the other systems. If one is not interested in topology and metric spaces, the extent of the provided libraries is mostly the same and these systems seem equally suitable for doing real analysis. (They are nowhere near providing what could be called modern analysis though.) So the choice of one system over another will mostly depend on personal preferences. It should be remembered though that Mizar is based on a set-theoretical setting while all the other ones are type-based, so they might be more user-friendly. Note also that the Coq standard library for real numbers has not evolved much since its inception, so it is a bit outdated now; formalizations for PVS, HOL4, HOL Light, Isabelle/HOL, and ProofPower-HOL, have more closely tracked the evolutions of the underlying system. With respect to automation, PVS, HOL Light, Isabelle/HOL, HOL4, and Coq (standard library), are of similar strength: they have powerful procedures for arithmetic, but such procedures are somehow lacking when it comes to analysis. Finally, except for Coq, all the standard libraries we have surveyed provide a formalization of complex numbers, which might be of importance depending on the kind of analysis development the user wants to tackle.

In the course of this survey, we encountered a few general issues that are worth reporting, as users are likely to stumble upon them. First of all, some standard libraries are in most parts poorly documented and one has to dive into the code of these systems to have a grasp of their extent. This may also have caused us to miss some hidden features of some systems. In particular, one can regret that, in this day and age, so few libraries have some browsable content online, *i.e.* an easy way to navigate from theorem statements to concept definitions without relying on dedicated tools.

More importantly, it appears that some formalizations were developed in isolation, without any concern for the applications that might be built upon them later. As a consequence, while elegant, some developments might prove difficult to use in practice, due to some missing lemmas or the lack of specific methods of automation. A lesson for the future would be that applications should drive the development of formalizations and libraries, rather than the opposite. Yet developers should strive to keep some homogeneity and consistency in their formalizations, *e.g.* naming.

In this survey, we have restricted ourselves to a small set of systems and libraries and are far from exhaustive. One can find numerous other developments formalizing real numbers out there. For instance, Constable et al. (1986) and Bickford (2008) provide formalizations for Nuprl. Jones (1993) shows how to complete metric spaces in LEGO, and thus how to build the real numbers from the set of rational numbers. Ciaffaglione and Di-Gianantonio (2006) formalize real numbers in Coq as infinite stream of digits in $\{-1, 0, 1\}$. Using Coq too, Cohen (2012) presents a formalization of algebraic numbers.

There are also works that extend the libraries presented in this survey. For instance, Avigad and Donnelly (2004) formalize the O notation in Isabelle/HOL. Richter (2004) defines the Lebesgue integral and uses it for reasoning on probabilistic algorithms in Isabelle/HOL. Lester (2007) develops a theory of topology in PVS. Julien and Paşca (2009)

combine Coq's standard library with computable reals defined as infinite streams. Boldo et al. (2012) extend Coq's standard library to simplify differentiation and integration.

To conclude this survey, we mention some large formal developments that were built thanks to the systems we presented. First, there are all the formalizations related to the verification of floating-point programs. Since floating-point arithmetic is used in practice as an approximation of real arithmetic, formal verification ends up relying on the latter. Most of the works have been performed with HOL Light, *e.g.* Harrison (1997), and Coq, *e.g.* Boldo and Melquiond (2011). There have also been some developments in ACL2, *e.g.* Moore et al. (1998), but it should be noted that they took great care to avoid irrational numbers and thus used plain ACL2 only.

Another kind of application is the study of ordinary or partial differential equations. For ordinary differential equations, proofs about one-step methods and the Euler method have been done in Isabelle/HOL with an executable specification (Immler and Hölzl 2012). Still for ordinary differential equations, a computable Picard operator has been proved in C-CoRN/MathClasses (Makarov and Spitters 2013). Regarding partial differential equations, a major development about the numerical resolution of the wave equation was done in standard Coq (Boldo et al. 2013).

Possibly one of the largest developments based on real analysis is the Flyspeck project; this is a HOL Light formalization of the theorem about dense sphere packings (Hales 2012, Solovyev and Hales 2013). Finally, one should note all the PVS works on verifying critical systems for avionics, *e.g.* Narkawicz et al. (2012).

7. Acknowledgments

The authors are grateful to Assia Mahboubi, Micaela Mayero, César Muñoz, Bas Spitters, and Makarius Wenzel, for their numerous and helpful comments on this survey. We are also grateful to the anonymous reviewers for their constructive remarks.

References

Arthan, R. D. (2001), An irrational construction of $\mathbb{R}$ from $\mathbb{Z}$, *in* R. J. Boulton and P. B. Jackson, eds, 'Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2001)', Vol. 2152 of *Lecture Notes in Computer Science*, Edinburgh, UK, pp. 43–58.

Arthan, R. D. (2012), *Mathematical Case Studies: Basic Analysis*, Lemma 1 Ltd.

Arthan, R. D. and King, D. (1996), 'Development of practical verification tools', *ICL Systems Journal* **11**(1), 106–122.

Avigad, J. and Donnelly, K. (2004), Formalizing O notation in Isabelle/HOL, *in* D. Basin and M. Rusinowitch, eds, 'Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR 2004)', Vol. 3097 of *Lecture Notes in Computer Science*, pp. 357–371.

Bertot, Y. and Castéran, P. (2004), *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*, Texts in Theoretical Computer Science, Springer Verlag.

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

Bickford, M. (2008), 'Formalizing Constructive Analysis in Nuprl', Computing Science Department, Cornell University.
URL: *http://www.nuprl.org/MathLibrary/ConstructiveAnalysis/*

Boldo, S., Clément, F., Filliâtre, J.-C., Mayero, M., Melquiond, G. and Weis, P. (2013), 'Wave equation numerical resolution: a comprehensive mechanized proof of a C program', *Journal of Automated Reasoning* **50**(4), 423–456.

Boldo, S., Lelay, C. and Melquiond, G. (2012), Improving real analysis in Coq: a user-friendly approach to integrals and derivatives, *in* C. Hawblitzel and D. Miller, eds, 'Proceedings of the 2nd International Conference on Certified Programs and Proofs (CPP 2012)', Vol. 7679 of *Lecture Notes in Computer Science*, Kyoto, Japan, pp. 289–304.

Boldo, S. and Melquiond, G. (2011), Flocq: A unified library for proving floating-point algorithms in Coq, *in* E. Antelo, D. Hough and P. Ienne, eds, 'Proceedings of the 20th IEEE Symposium on Computer Arithmetic (ARITH 20)', Tübingen, Germany, pp. 243–252.

Butler, R. W. (2009), 'Formalization of the integral calculus in the PVS theorem prover', *Journal of Formalized Reasoning* **2**(1), 1–26.

Byliński, C. (1990), 'Functions and their basic properties', *Journal of Formalized Mathematics* **1**(1), 55–65.

Chaieb, A. (2006), Verifying mixed real-integer quantifier elimination, *in* U. Furbach and N. Shankar, eds, 'Proceedings of the 3rd International Joint on Automated Reasoning (IJCAR 2006)', Vol. 4130 of *Lecture Notes in Computer Science*, Seattle, WA, USA, pp. 528–540.

Chaieb, A. (2008), Parametric linear arithmetic over ordered fields in Isabelle/HOL, *in* S. Autexier, J. Campbell, J. Rubio, V. Sorge, M. Suzuki and F. Wiedijk, eds, 'Intelligent Computer Mathematics', Vol. 5144 of *Lecture Notes in Artificial Intelligence*, Birmingham, UK, pp. 246–260.

Chaieb, A. and Wenzel, M. (2007), Context aware calculation and deduction – Ring equalities via Gröbner bases in Isabelle, *in* M. Kauers, M. Kerber, R. Miner and W. Windsteiger, eds, 'Proceedings of the 14th Symposium Calculemus 2007, Towards Mechanized Mathematical Assistants', Vol. 4573 of *Lecture Notes in Artificial Intelligence*, Hagenberg, Austria, pp. 27–39.

Ciaffaglione, A. and Di-Gianantonio, P. (2006), 'A certified, corecursive implementation of exact real numbers', *Theoretical Computer Science* **351**(1), 39–51.

Cohen, C. (2012), Formalized algebraic numbers: construction and first order theory, PhD thesis, École Polytechnique.

Constable, R. L., Allen, S. F., Allen, S. F., Bromley, H. M., Cleaveland, W. R., Cremer, J. F., Harper, R. W., Howe, D. J., Knoblock, T. B., Mendler, N. P., Panangaden, P., Smith, S. F., Sasaki, J. T. and Smith, S. F. (1986), 'Implementing mathematics

with the Nuprl proof development system', Computing Science Department, Cornell University.

Cowles, J. R. and Gamboa, R. (2010), Using a first order logic to verify that some set of reals has no Lesbegue measure, *in* M. Kaufmann and L. C. Paulson, eds, 'Proceeding of the 1st International Conference of Interactive Theorem Proving (ITP 2010)', Vol. 6172 of *Lecture Notes in Computer Science*, Edinburgh, UK, pp. 25–34.

Cruz-Filipe, L. (2003), A constructive formalization of the fundamental theorem of calculus, *in* 'Proceedings of the International Conference on Types for Proofs and Programs (TYPES'02)', Vol. 2646 of *Lecture Notes in Computer Science*, Berg en Dal, The Netherlands, pp. 108–126.

Cruz-Filipe, L. (2004), Constructive Real Analysis: a Type-Theoretical Formalization and Applications, PhD thesis, University of Nijmegen.

Cruz-Filipe, L., Geuvers, H. and Wiedijk, F. (2004), C-CoRN: the constructive Coq repository at Nijmegen, *in* A. Asperti, G. Bancerek and A. Trybulec, eds, 'Proceedings of the 3rd International Conference of Mathematical Knowledge Management (MKM 2004)', Vol. 3119 of *Lecture Notes in Computer Science*, pp. 88–103.

Daumas, M., Lester, D. and Muñoz, C. (2009), 'Verified real number calculations: A library for interval arithmetic', *IEEE Transactions on Computers* **58**(2), 226–237.

Delahaye, D. and Mayero, M. (2001), Field, une procédure de décision pour les nombres réels en Coq, *in* P. Castéran, ed., 'Journées francophones des langages applicatifs (JFLA'01)', pp. 33–48.

Desmettre, O. (2002). Coq standard library – RiemannInt.

Dutertre, B. (1996), Elements of mathematical analysis in PVS, *in* J. von Wright, J. Grundy and J. Harrison, eds, 'Proceedings of the 9th International Conference Theorem Proving in Higher Order Logics (TPHOLs 1996)', Vol. 1125 of *Lecture Notes in Computer Science*, Turku, Finland, pp. 141–156.

Endou, N. and Korniłowicz, A. (1999), 'The definition of the Riemann definite integral and some related lemmas', *Journal of Formalized Mathematics* **8**(1), 93–102.

Endou, N., Wasaki, K. and Shidama, Y. (2001), 'Definition of integrability for partial functions from $\mathbb{R}$ to $\mathbb{R}$ and integrability for continuous functions', *Formalized Mathematics* **9**(**2**), 281–284.

Fleuriot, J. (2000), On the mechanization of real analysis in Isabelle/HOL, *in* M. Aagaard and J. Harrison, eds, 'Proceeding of the 13th International Conference of Theorem Proving in Higher Order Logics (TPHOLs 2000)', Vol. 1869 of *Lecture Notes in Computer Science*, Portland, OR, USA, pp. 145–161.

Gamboa, R. (2000), Continuity and differentiability in ACL2, *in* M. Kaufmann, P. Manolios and J. S. Moore, eds, 'Computer-Aided Reasoning: ACL2 Case Studies', Advances in Formal Methods, pp. 301–316.

Gamboa, R. and Cowles, J. R. (2009), Inverse functions in ACL2(r), *in* 'Proceedings of the 8th International Workshop on the ACL2 Theorem Prover and its Applications', pp. 57–61.

Gamboa, R., Cowles, J. R. and Kuzmina, N. (2004), Axiomatic events in ACL2(r): A story of defun, defun-std, and encapsulate, *in* 'Proceedings of the 5th International Workshop on the ACL2 Theorem Prover and its Applications', Austin, TX, USA.

Gamboa, R. and Kaufmann, M. (2001), 'Nonstandard analysis in ACL2', *Journal of Automated Reasoning* **27**(4), 323–351.

Garillot, F., Gonthier, G., Mahboubi, A. and Rideau, L. (2009), Packaging mathematical structures, *in* S. Berghofer, T. Nipkow, C. Urban and M. Wenzel, eds, 'Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)', Vol. 5674 of *Lecture Notes in Computer Science*, München, Germany, pp. 327–342.

Geuvers, H. and Niqui, M. (2002), Constructive reals in Coq: Axioms and categoricity, *in* P. Callaghan, Z. Luo, J. McKinna and R. Pollack, eds, 'Selected Papers of the International Workshop on Types for Proofs and Programs (TYPES 2000)', Vol. 2277 of *Lecture Notes in Computer Science*, Durham, UK, pp. 79–95.

Gordon, M. J. C. and Melham, T. F. (1993), *Introduction to HOL: a theorem proving environment for higher order logic*, Cambridge University Press.

Gottliebsen, H. (2000), Transcendental functions and continuity checking in PVS, *in* M. Aagaard and J. Harrison, eds, 'Proceedings of the 13th International Conference Theorem Proving in Higher Order Logics (TPHOLs 2000)', Vol. 1869 of *Lecture Notes in Computer Science*, Portland, OR, USA, pp. 197–214.

Grégoire, B. and Mahboubi, A. (2005), Proving equalities in a commutative ring done right in Coq, *in* J. Hurd and T. Melham, eds, 'Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2005)', Vol. 3603 of *Lecture Notes in Computer Science*, Oxford, UK, pp. 98–113.

Guard, J. R., Oglesby, F. C., Bennett, J. H. and Settle, L. G. (1969), 'Semi-automated mathematics', *Journal of the ACM* **16**, 49–62.

Hales, T. (2012), *Dense Sphere Packings: A Blueprint for Formal Proofs*, London Mathematical Society Lecture Note Series, Cambridge University Press.

Harrison, J. (1994), 'Constructing the real numbers in HOL', *Formal Methods in System Design* **5**(1–2), 35–59.

Harrison, J. (1996), Proof style, *in* 'Proceedings of the International Workshop on Types for Proofs and Programs (TYPES'96)', Vol. 1512 of *Lecture Notes in Computer Science*, Aussois, France, pp. 154–172.

Harrison, J. (1997), Floating point verification in HOL Light: the exponential function, Technical Report 428, University of Cambridge Computer Laboratory.

Harrison, J. (1998), *Theorem Proving with the Real Numbers*, Springer-Verlag.

Harrison, J. (2005), A HOL theory of Euclidean space, *in* J. Hurd and T. Melham, eds, 'Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2005)', Vol. 3603 of *Lecture Notes in Computer Science*, Oxford, UK, pp. 114–129.

Harrison, J. (2006), Towards self-verification of HOL Light, *in* U. Furbach and N. Shankar, eds, 'Proceedings of the 3rd International Joint Conference (IJCAR 2006)', Vol. 4130 of *Lecture Notes in Computer Science*, Seattle, WA, USA, pp. 177–191.

Harrison, J. (2007), Verifying nonlinear real formulas via sums of squares, *in* K. Schneider and J. Brandt, eds, 'Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2007)', Vol. 4732 of *Lecture Notes in Computer Science*, Kaiserslautern, Germany, pp. 102–118.

Harrison, J. (2009), HOL Light: An overview, *in* S. Berghofer, T. Nipkow, C. Urban and M. Wenzel, eds, 'Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)', Vol. 5674 of *Lecture Notes in Computer Science*, München, Germany, pp. 60–66.

Harrison, J. (2013), 'The HOL Light theory of Euclidean space', *Journal of Automated Reasoning* **50**, 173–190.

HOL4 development team (2012), The HOL System LOGIC, Technical report, University of Cambridge, NICTA and University of Utah.

Hölzl, J., Immler, F. and Huffman, B. (2013), Type classes and filters for mathematical analysis in Isabelle/HOL, *in* S. Blazy, C. Paulin-Mohring and D. Pichardie, eds, 'Proceedings of the 4th International Conference on Interactive Theorem Proving (ITP 2013)', Vol. 7998 of *Lecture Notes in Computer Science*, Rennes, France, pp. 279–294.

Hunt, Jr., W. A., Krug, R. B. and Moore, J. (2003), Linear and nonlinear arithmetic in ACL2, *in* D. Geist and E. Tronci, eds, 'Proceedings of the 12th IFIP WG 10.5 Conference on Correct Hardware Design and Verification Methods (CHARME 2003)', Vol. 2860 of *Lecture Notes in Computer Science*, L'Aquila, Italy, pp. 319–333.

Hurd, J. (2011), The OpenTheory standard theory library, *in* M. Bobaru, K. Havelund, G. J. Holzmann and R. Joshi, eds, 'Proceedings of the 3rd International Symposium on NASA Formal Methods (NFM 2011)', Vol. 6617 of *Lecture Notes in Computer Science*, Pasadena, CA, USA, pp. 177–191.

Hölzl, J. and Heller, A. (2011), Three chapters of measure theory in Isabelle/HOL, *in* M. Eekelen, H. Geuvers, J. Schmaltz and F. Wiedijk, eds, 'Proceedings of the 2nd International Conference of Interactive Theorem Proving (ITP 2011)', Vol. 6898 of *Lecture Notes in Computer Science*, Berg en Dal, The Netherlands, pp. 135–151.

Immler, F. and Hölzl, J. (2012), Numerical analysis of ordinary differential equations in Isabelle/HOL, *in* L. Beringer and A. Felty, eds, 'Proceedings of the 3rd International Conference on Interactive Theorem Proving (ITP 2012)', Vol. 7406 of *Lecture Notes in Computer Science*, Princeton, NJ, USA, pp. 377–392.

Jones, C. (1993), Completing the rationals and metric spaces in LEGO, *in* G. Huet and G. Plotkin, eds, 'Papers presented at the second annual Workshop on Logical environments', Cambridge University Press, Edinburgh, UK, pp. 297–316.

Julien, N. and Paşca, I. (2009), Formal verification of exact computations using Newton's method, *in* S. Berghofer, T. Nipkow, C. Urban and M. Wenzel, eds, 'Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)', Vol. 5674 of *Lecture Notes in Computer Science*, München, Germany, pp. 408–423.

Jutting, L. S. v. B. (1977), Checking Landau's "Grundlagen" in the AUTOMATH System, PhD thesis, Eindhoven University of Technology.

Kaliszyk, C. and O'Connor, R. (2009), 'Computing with classical real numbers', *Journal of Automated Reasoning* **2**(1), 27—-29.

Kaufmann, M., Moore, J. S. and Manolios, P. (2000), *Computer-Aided Reasoning: An Approach*, Kluwer Academic Publishers, Norwell, MA, USA.

Kotowicz, J. (1990*a*), 'Convergent sequences and the limit of sequences', *Journal of Formalized Mathematics* **1**(2), 273–275.

Kotowicz, J. (1990*b*), 'Real sequences and basic operations on them', *Journal of Formalized Mathematics* **1**(2), 269–272.

Kotowicz, J. (1991*a*), 'The limit of a real function at a point', *Journal of Formalized Mathematics* **2**(1), 71–80.

Kotowicz, J. (1991*b*), 'The limit of a real function at infinity', *Journal of Formalized Mathematics* **2**, 17–28.

Kotowicz, J. (1991*c*), 'One-side limits of a real function at a point', *Journal of Formalized Mathematics* **2**(1), 29–40.

Krebbers, R. and Spitters, B. (2013), 'Type classes for efficient exact real arithmetic in Coq', *Logical Methods in Computer Science* **9**(1:1), 1–27.

Lester, D. R. (2007), Topology in PVS: continuous mathematics with applications, *in* 'Proceedings of the 2nd workshop on Automated Formal Methods (AFM '07)', Atlanta, GA, USA, pp. 11–20.

Lester, D. R. (2008), Real number calculations and theorem proving, *in* O. A. Mohamed, C. A. Muñoz and S. Tahar, eds, 'Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008)', Vol. 5170 of *Lecture Notes in Computer Science*, pp. 215–229.

Makarov, E. and Spitters, B. (2013), The Picard algorithm for ordinary differential equations in Coq, *in* S. Blazy, C. Paulin-Mohring and D. Pichardie, eds, 'Proceeding of the 4th International Conference of Interactive Theorem Proving (ITP 2013)', Vol. 7998 of *Lecture Notes in Computer Science*, Rennes, France, pp. 463–468.

Manolios, P. and Moore, J. S. (2003), 'Partial functions in ACL2', *Journal of Automated Reasoning* **31**(2), 107–127.

Mayero, M. (2001), Formalisation et automatisation de preuves en analyses réelle et numérique, PhD thesis, Université Paris VI.

Mayero, M. (2012), Problèmes critiques et preuves formelles, Habilitation à diriger des recherches, Université Paris 13.

McLaughlin, S. and Harrison, J. (2005), A proof-producing decision procedure for real arithmetic, *in* R. Nieuwenhuis, ed., 'Proceedings of the 20th International Conference on Automated Deduction (CADE 20)', Vol. 3632 of *Lecture Notes in Computer Science*, Tallinn, Estonia, pp. 295–314.

Mhamdi, T., Hasan, O. and Tahar, S. (2010), On the formalization of the Lebesgue integration theory in HOL, *in* M. Kaufmann and L. C. Paulson, eds, 'Proceedings of the 1st International Conference of Interactive Theorem Proving (ITP 2010)', Vol. 6172 of *Lecture Notes in Computer Science*, Edinburgh, UK, pp. 387–402.

Monniaux, D. and Corbineau, P. (2011), On the generation of Positivstellensatz witnesses in degenerate cases, *in* M. Van Eekelen, H. Geuvers, J. Schmaltz and F. Wiedijk, eds, 'Proceedings of the 2nd International Conference on Interactive Theorem Proving (ITP 2011)', Vol. 6898 of *Lecture Notes in Computer Science*, Berg en Dal, The Netherlands, pp. 249–264.

Moore, J. S., Lynch, T. and Kaufmann, M. (1998), 'A mechanically checked proof of the correctness of the kernel of the AMD5K86 floating point division algorithm', *IEEE Transactions on Computers* **47**(9), 913–926.

Muñoz, C. and Mayero, M. (2001), Real automation in the field, Technical Report NASA/CR-2001-211271, ICASE.

Muñoz, C. and Narkawicz, A. (2013), 'Formalization of a Bernstein polynomials and applications to global optimization', *Journal of Automated Reasoning* **51**(2), 151–196.

Narkawicz, A., Muñoz, C. and Dowek, G. (2012), 'Provably correct conflict prevention bands algorithms', *Science of Computer Programming* **77**(10–11), 1039–1057.

Naumowicz, A. and Korniłowicz, A. (2009), A brief overview of Mizar, *in* S. Berghofer, T. Nipkow, C. Urban and M. Wenzel, eds, 'Proceedings of the 22th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)', Vol. 5674 of *Lecture Notes in Computer Science*, München, Germany, pp. 67–72.

Nipkow, T., Paulson, L. C. and Wenzel, M. (2002), *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, Vol. 2283 of *Lecture Notes in Computer Science*, Springer.

O'Connor, R. (2007), 'A monadic, functional implementation of real numbers', *Mathematical Structures in Computer Science* **17**, 129–159.

O'Connor, R. (2008), Certified exact transcendental real number computation in Coq, *in* O. A. Mohamed, C. Muñoz and S. Tahar, eds, 'Proceedings of the 21th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008)', Vol. 5170 of *Lecture Notes in Computer Science*, Montreal, Canada, pp. 246–261.

Owre, S., Rushby, J. M., and Shankar, N. (1992), PVS: A prototype verification system, *in* D. Kapur, ed., 'Proceedings of the 11th International Conference on Automated Deduction (CADE 11)', Vol. 607 of *Lecture Notes in Artificial Intelligence*, Saratoga, NY, pp. 748–752.

Owre, S. and Shankar, N. (2003), The PVS Prelude Library, Technical report, CSL Technical Report SRI-CSL-03-01.

O'Connor, R. and Spitters, B. (2010), 'A computer-verified monadic functional implementation of the integral', *Theoretical Computer Science* **411**(37), 3386–3402.

Perkowska, B. (1992), 'Functional sequence from a domain to a domain', *Journal of Formalized Mathematics* **3**(1), 17–21.

Pottier, L. (2008), 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, eds, 'Knowledge Exchange: Automated Provers and Proof Assistants', CEUR Workshop Proceedings, Doha, Qatar, pp. 67–76.

ProofPower development team (2006), Proofpower - HOL reference manual, Technical report, Lemma 1 Ltd.

Raczkowski, K. (1991), 'Integer and rational exponents', *Journal of Formalized Mathematics* **2**(1), 125–130.

Raczkowski, K. and Nedzusiak, A. (1991*a*), 'Real exponents and logarithms', *Journal of Formalized Mathematics* **2**(2), 213–216.

Raczkowski, K. and Nedzusiak, A. (1991*b*), 'Series', *Journal of Formalized Mathematics* **2**(4), 449–452.

Raczkowski, K. and Sadowski, P. (1990*a*), 'Real function continuity', *Journal of Formalized Mathematics* **1**(4), 787–791.

Raczkowski, K. and Sadowski, P. (1990*b*), 'Real function differentiability', *Journal of Formalized Mathematics* **1**(4), 797–801.

Reid, P. and Gamboa, R. (2011), Automatic differentiation in ACL2, *in* M. Eekelen, H. Geuvers, J. Schmaltz and F. Wiedijk, eds, 'Proceeding of the 2nd International

Conference of Interactive Theorem Proving (ITP 2011)', Vol. 6898 of *Lecture Notes in Computer Science*, Berg en Dal, The Netherlands, pp. 312–324.

Richter, S. (2004), Formalizing integration theory with an application to probabilistic algorithms, *in* K. Slind, A. Bunker and G. Gopalakrishnan, eds, 'Proceedings of the 17th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2004)', Vol. 3223 of *Lecture Notes in Computer Science*, Park City, UT, USA, pp. 271–286.

Rushby, J., Owre, S. and Shankar, N. (1998), 'Subtypes for specifications: Predicate subtyping in PVS', *IEEE Transactions on Software Engineering* **24**(9), 709–720.

Shi, Z., Gu, W., Li, X., Guan, Y., Ye, S., Zhang, J. and Wei, H. (2013), 'The gauge integral theory in HOL4', *Journal of Applied Mathematics* .

Solovyev, A. and Hales, T. C. (2013), Formal verification of nonlinear inequalities with taylor interval approximations, *in* G. Brat, N. Rungta and A. Venet, eds, 'Proceedings of the 5th NASA Formal Methods Symposium (NFM 2013)', Vol. 7871 of *Lecture Notes in Computer Science*, pp. 383–397.

Spitters, B. and van der Weegen, E. (2011), 'Type classes for mathematics in type theory', *Mathematical Structures of Computer Science* **21**, 1–31.

Trybulec, A. (1993), Some features of the Mizar language, *in* 'Proceedings of the ESPRIT Workshop', Torino, Italy.

Trybulec, A. (1998), 'Non negative real numbers. Part I', *Journal of Formalized Mathematics* Addenda.

Vito, B. L. D. (2003), Strategy-enhanced interactive proving and arithmetic simplification for PVS, *in* 'Proceedings of STRATA 2003, First International Workshop on Design and Application of Strategies/Tactics in Higher Order Logics', number CP-2003-212448, NASA, Rome, Italy.

Wenzel, M. (2002), Isabelle/Isar – a versatile environment for human-readable formal proof documents, PhD thesis, Institut für Informatik, Technische Universität München.

Wenzel, M. and Wiedijk, F. (2002), 'A comparison of the mathematical proof languages Mizar and Isar', *Journal of Automated Reasoning* **29**, 389–411.

Wiedijk, F. (2006), *The Seventeen Provers of the World*, Vol. 3600 of *Lecture Notes in Artificial Intelligence*, Springer. Foreword by Dana S. Scott.

Wiedijk, F. (2009), 'Statistics on digital libraries of mathematics', *Studies in Logic, Grammar and Rhetoric* **18**(31), 137–151. Computer Reconstruction of the Body of Mathematics.