Formalizing Mathematics in Coq

Luís Cruz-Filipe

University of Nijmegen
Centro de Lógica e Computação

January 4, 2002

Overview

1. Introduction

2. Formalizing Mathematics: How and Why

3. Proof Assistants—Underlying Theory

4. Coq: Specific Characteristics

5. Towards a Formalization of Real Analysis

6. Conclusions & Future Work

Formalizing Mathematics

Why?

- Higher reliability for proofs
- Potential auxiliary tool in investigation
- Applications

How?

- Proof Assistants (Coq) which interactively generate proofs which are easy to check and perform computations.

Typed λ-calculus: terms and types.

- $\mathbb{V}$ is a set of *type variables*

- $\{x_i | i \in \mathbb{N}\}$ is a countable set of *term variables*

$$\begin{aligned} \mathbb{T} &:= \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \\ \Lambda &:= x_i \mid \lambda x_i : \mathbb{T}.\Lambda \mid \Lambda\Lambda \end{aligned}$$

$M : A$ means that the term M has type A.

A *context* Γ is a set of judgements of the form $M : A$.

Types of terms are inductively defined:

- if $M : A \in \Gamma$ then $\Gamma \vdash M : A$
- if $\Gamma, x : A \vdash M : B$ then $\Gamma \vdash (\lambda x : A.M) : A \rightarrow B$
- if $\Gamma \vdash M : A \rightarrow B$ and $\Gamma \vdash N : A$ then $\Gamma \vdash (MN) : B$

A is said to be *inhabited* iff there exists some M such that $\vdash M : A$

Let A, B and C be type variables and define

$$
\begin{aligned}
\mathbf{S} &:= \lambda x{:}(A \to B \to C).\lambda y{:}A \to B.\lambda z{:}A.xz(yz) \\
\mathbf{K} &:= \lambda x{:}A.\lambda y{:}B.x
\end{aligned}
$$

Then:

$$
\begin{aligned}
&\vdash \mathbf{K} : A \to B \to A \\
&\vdash \mathbf{S} : (A \to B \to C) \to (A \to B) \to A \to C
\end{aligned}
$$

Viewing type variables as propositional variables and $\rightarrow$ as (intuitionistic) implication, we have that:

- the rules for typing correspond exactly to the natural deduction rules for implication introduction and elimination in the implicative fragment of intuitionistic propositional logic

- the types of $\mathbf{K}$ and $\mathbf{S}$ correspond to intuitionistic tautologies

We have both *correction* and *completeness*.

Pure Type Systems

A *Pure Type System* (PTS) is a triple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ where:

- $\mathcal{S}$ is a set
- $\mathcal{A} \subseteq \mathcal{S}^2$
- $\mathcal{R} \subseteq \mathcal{S}^3$

The elements of $\mathcal{S}$ are called *sorts*, the elements of $\mathcal{A}$ are called *axioms* and the elements of $\mathcal{R}$ are called *rules*. We will usually represent an axiom $\langle x, A \rangle$ by $x : A$ and abbreviate rules of the form $\langle s_1, s_2, s_2 \rangle$ to $\langle s_1, s_2 \rangle$.

Type assignment rules for PTS:

$$\text{(sort)} \qquad \vdash s_1 : s_2 \qquad s_1 : s_2 \in \mathcal{A}$$

$$\text{(var)} \qquad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad x \notin \Gamma$$

$$\text{(weak)} \qquad \frac{\Gamma \vdash A : s \qquad \Gamma \vdash M : C}{\Gamma, x : A \vdash M : C} \qquad x \notin \Gamma$$

$$\text{(product)}\quad \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_3} \quad \langle s_1, s_2, s_3 \rangle \in \mathcal{R}$$

$$\text{(abstraction)}\quad \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

$$\text{(application)}\quad \frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : B}{\Gamma \vdash M N : B\,[N/x]}$$

$$\text{(conversion)}\quad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B} \quad A =_R B$$

Important properties of PTS are:

Thinning: if $\Gamma \subseteq \Gamma'$ and $\Gamma \vdash M : A$ then $\Gamma' \vdash M : A$

Substitution: if $\Gamma_1, x : B, \Gamma_2 \vdash M : A$ and $\Gamma_1 \vdash N : B$ then $\Gamma_1, \Gamma_2 [N/x] \vdash M [N/x] : A [N/x]$

Strengthening: if $\Gamma_1, x : B, \Gamma_2 \vdash M : A$ and $x \notin FV(\Gamma_2, M, A)$ then $\Gamma_1, \Gamma_2 \vdash M : A$

Reduction: if $\Gamma \vdash M : A$ and $M \to_\beta^* N$ then $\Gamma \vdash N : A$

A PTS such that $\mathcal{A}$ and $\mathcal{R}$ are functions is called *functional*. Functional PTS enjoy the following property:

Uniqueness: if $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A =_\beta B$

A *morphism* between two PTS $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ and $\langle \mathcal{S}', \mathcal{A}', \mathcal{R}' \rangle$ is a function $f : \mathcal{S} \to \mathcal{S}'$ such that $f(\mathcal{A}) \subseteq \mathcal{A}'$ and $f(\mathcal{R}) \subseteq \mathcal{R}'$.

Morphisms preserve β-reduction, the diamond property and strong normalization.

$\langle \{*\}, \{* : *\}, \{\langle *, *, * \rangle\} \rangle$ is a terminal object in the category of all PTS.

An important class of PTS are those where we just have two sorts $*$ and $\Box$ and the single axiom $* : \Box$. By combining these sorts in all possible ways to generate rules of the form $\langle s_1, s_2, s_2 \rangle$ we get eight PTS generally known as the *Lambda Cube*, which are usually presented in the following graphical way:

$$\begin{tikzcd}
 & \lambda\omega \arrow[rr] & & \lambda P\omega \\
\lambda 2 \arrow[ur] \arrow[rr] & & \lambda P2 \arrow[ur] & \\
 & \lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \lambda P\underline{\omega} \arrow[uu] \\
\lambda{\to} \arrow[uu] \arrow[ur] \arrow[rr] & & \lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

Inductive types and ι-reduction

Inductive $\mu : s :=$

$$\begin{array}{rcl} \mathsf{constr}_1 & : & \sigma_1^1(\mu) \to \ldots \to \sigma_{m_1}^1(\mu) \to \mu \\ & \vdots & \\ |\mathsf{constr}_n & : & \sigma_1^n(\mu) \to \ldots \to \sigma_{m_n}^n(\mu) \to \mu \end{array}$$

where each $\sigma_j^i(\mu)$ is of the form $A_1 \to A_k$ with μ not occurring in $A_1, \ldots, A_k$ and either X is μ or μ does not occur in X.

Inductive types come with induction and recursion principles.

Inductive $\texttt{nat}$: Type :=

$$\begin{array}{rcl} 0 & : & \texttt{nat} \\ |S & : & \texttt{nat} \to \texttt{nat} \end{array}$$

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma \vdash f_1 : A \quad \Gamma \vdash f_2 : \texttt{nat} \to A \to A}{\Gamma \vdash \mathsf{Rec}_{\texttt{nat}} f_1 f_2 : \texttt{nat} \to A} \, elim_1$$

$$\frac{\Gamma \vdash P : \texttt{nat} \to \mathsf{Prop} \quad \Gamma \vdash f_1 : P0 \quad \Gamma \vdash f_2 : \Pi x : \texttt{nat}.Px \to P(Sx)}{\Gamma \vdash \mathsf{Rec}_{\texttt{nat}} f_1 f_2 : \Pi x : \texttt{nat}.Px} \, elim_2$$

$$\begin{array}{rcl} \mathsf{Rec}_{\texttt{nat}} f_1 f_2 0 & \to_\iota & f_1 \\ \mathsf{Rec}_{\texttt{nat}} f_1 f_2 (St) & \to_\iota & f_2 t (\mathsf{Rec}_{\texttt{nat}} f_1 f_2 t) \end{array}$$

The Calculus of Inductive Constructions

$$
\begin{aligned}
\mathcal{S} &= \{\mathbf{Set}, \mathbf{Prop}, \mathbf{Type}(i) | i \in \mathbb{N}\} \\
\mathcal{A} &= \{\mathbf{Set} : \mathbf{Type}(0), \mathbf{Prop} : \mathbf{Type}(0), \mathbf{Type}(i) : \mathbf{Type}(i+1) | i \in \mathbb{N}\} \\
\mathcal{R} &= \{\langle s_1, s_2 \rangle | s_1 \in \{\mathbf{Set}, \mathbf{Prop}\} \text{ or } s_1 \in \{\mathbf{Set}, \mathbf{Prop}\}\} \\
& \bigcup \{\langle s_1, s_2, s_3 \rangle | s_i := \mathbf{Type}(n_i), n_1 \leq n_3 \text{ and } n_2 \leq n_3\}
\end{aligned}
$$

- no η-reduction
- restrictions to elimination over inductive types (due to consistency problems)

In Coq:

- $\lambda x{:}A.B$ is written as `[x:A]B`
- $\Pi x{:}A.B$ is written as `(x:A)B`
- $\mathbf{Type}(i)$ is written simply as `Type`
- δ-reduction
- special inductive type: `Record`

Implementation options

- Constructive mathematics

- η-reduction vs. setoids

- Coercions

- `Set`-based logic

My work

Goals: Formalization of main concepts and results of Calculus in one real variable:

- Taylor's Theorem
- Derivation rules
- Fundamental Theorem of Calculus

Environment: Work previously done in order to prove the Fundamental Theorem of Algebra, which already included:

- A construction of the reals as a complete ordered field satisfying the Archimedean axiom

- Formalized definitions of most common operations and constructions on the real numbers (algebraic operations, absolute value, maximum, Cauchy sequences, limit)

- Formalized proofs of the main properties of these operations

- Formalized notions of real valued (total) functions and pointwise continuity

Problems:

- No obvious concept of partial function
- Definitions depending on proofs
- Classical definitions won't work
- Little automation
- Need of "unfolding" lemmas

Example: from Rolle's Theorem to the Mean Law

```
(a,b:IR; f:(CSetoid_fun (subset (compact a b)) IR);
        diffF:(diffble_I a b f))
        (f A) [=] (f B)
        ->(e:IR)(Zero[<]e)
           ->{x:(subset (compact a b))
                & ((AbsIR
                     (projS1 ?? diffF x)) [<=] e)}
```

$\rightarrow$ can only be applied to the exact elimination of the existential quantifier

Generalization:

```
(a,b:IR; f,h:(CSetoid_fun (subset (compact a b)) IR))
    (f A) [=] (f B)
    ->(derivative_I a b f h)
    ->(e:IR)
       (Zero[<]e)
       ->{x:(subset (compact a b)) & ((AbsIR (h x)) [<=] e)}
```

We now want to prove the Mean Law:

```
Variables a,b:IR.
Local I:=(compact a b).

Local A,B.

Variable f:(CSetoid_fun (subset I) IR).

Hypothesis diffF:(diffble_I ?? f).

Local f':=(projS1 ?? diffF).

Lemma Mean_Law : (e:IR)(Zero[<]e)-> {x:(subset I) &
  (AbsIR ((f B)[-](f A))[-](f' x)[*](b[-]a))[<=]e}.
```

Technique: define a function $h : [a, b] \to \mathbb{R}$ such that

$$h(x) = (x - a)(f(b) - f(a)) - f(x)(b - a)$$

and apply Rolle's Theorem.

How much of this proof can be done automatically?

- prove that $h(a) = h(b)$
- compute $h'(x)$
- prove that $h'(x) = f(b) - f(a) - f'(x)(b - a)$

Reflection Tactics: `Rational` and `New_Deriv`

Motivation: work syntactically.

`Rational`: prove that two elements of an arbitrary field are equal.

`New_Deriv`: prove that one function f' is the derivative of another function f.

The `New_Deriv` Tactic

Inductive type $\mathcal{RF}$ of "restricted functions":

- constant and identity functions are in $\mathcal{RF}$;
- differentiable functions are in $\mathcal{RF}$
- synctatical expressions built up from restricted functions using $+$, $-$ and $*$ are in $\mathcal{RF}$

$\rightarrow$ There is a trivial mapping $[\![\cdot]\!]$ from $\mathcal{RF}$ into the class of functions from $[a, b]$ to $\mathbb{R}$.

$\rightarrow$ In $\mathcal{RF}$ we can inductively define a syntactical derivative function $'$ satisfying $[\![f']\!] = [\![f]\!]'$.

Goal: an expression of the form `(derivative_I a b f f')`.

Arguments: none.

Steps:

1. Determine an r such that $[\![r]\!] = f$ and substitute $[\![r]\!]$ for f in the goal
2. Calculate r'
3. Check that $[\![r']\!] = f'$

```
Tactic Definition New_Deriv :=
  Match Context With
    [|-(derivative_I ?1 ?2 ?3 ?4)] -> Let r=(ifunct_to_restr ?3) In
      Apply derivative_wdl with (restr_to_ifunct a b r); [
        Intro; Simpl; Algebra
        | Apply derivative_wdr with (restr_deriv a b r); [
          Intro; Simpl; Try Rational
          | Apply deriv_restr]].
```

Drawbacks:

- the equality proofs are not guaranteed to succeed; in this case, subgoals are left for the user to prove

Future Work

- Generalizing Rolle's and Taylor's Theorems to arbitrary partial functions

- Optimizing the tactics for automatic differentiation

- Theory of Integration

- Fundamental Theorem of Calculus