Formal Proofs in Coq: Kantorovitch’s Theorem

Ioana Pașca

University of Nice - Sophia Antipolis
INRIA Sophia Antipolis
PhD thesis advisor: Yves BERTOT

July 9, 2008
Outline

Proof Assistants

Coq

Verifying numerical algorithms

Formalizing mathematics
Proof assistant

- proof checker + proof-development system
- but, not a theorem prover

Motivation:
- increase reliability of mathematical proofs

Based on:
- a logic (classical/intuitionistic; first order/higher order . . .) and
- set theory or
- type theory

Example:
- based on set theory (Tarski - Grothendieck): Mizar
- based on type theory: HOL, Isabelle, Coq, ACL2, PVS, Agda, Lego, Nurpl, Minlog etc.
Why type theory?

a powerful formal system that captures

- *computation* (via the inclusion of functional programs written in typed $\lambda$-calculus),
- *proof* (via the “formulas as types embedding”, where types are viewed as propositions and terms as proofs)

Decidability of type checking = core of the type-theoretic theorem proving

\[
\Gamma \vdash \text{Type}_\Gamma(p) = A
\]
Applications

in mathematics

- complicated or complex problems:
  - four color theorem (G. Gonthier)
  - Kepler conjecture and T. Hales proof (Flyspeck project)

in computer science

- software and hardware verification
• **Calculus of Inductive Constructions**
  • dependent types
  • inductive types

• **Intuitionistic, Higher-order Logic**

• **Presence of Proof Objects**: the script generates and stores a term that is isomorphic to a proof that can be checked on independent/simple proof checker. $\implies$ high reliability.

• **Poincaré Principle** There is a distinction between *computations* and *proofs*; computations do not require a proof. (E.g. $1+0 = 1$ does not require a proof.)

• structurally well-founded recursion $\implies$ termination
Limits of Coq?

Marelle Team, INRIA, April 2008:

- limitis of pure functional programming: no computational effects (side effects, interactive input/output, exceptions,..);
- proof checker and not prover (2 researchers);
- syntactic restrictions: difficult to have different views/representations of one object;
- constructive logic ;
- structural recursion, guardedness...;
- higher-order unification;
- deciding guardedness;
- need for a better organised documentation.
What is the one best thing about Coq?

Marelle Team, INRIA, April 2008:

- mathematics and programming together; compute and prove simultaneously; \(\implies\) Research in Coq (3 researchers);
- dependent types;
- type theory \(\implies\) formal rigour;
- implicit arguments, type inference;
- extraction;
- replication of proofs;
- simple, uniform notation.
Successfull applications of Coq (http://coq.inria.fr/)

Mathematics

- Geometry,
- Set Theory,
- Algebra,
- Number theory,
- Category Theory,
- Domain theory,
- Real analysis and Topology,
- Probabilities.
Successfull applications of Coq (http://coq.inria.fr/)

Mathematics

- Geometry,
- Set Theory,
- Algebra,
- Number theory,
- Category Theory,
- Domain theory,
- Real analysis and Topology,
- Probabilities.

Computer Science

- Infinite Structures,
- Pr. Lang.: Data Types and Data Structures;
- Pr. Lang.: Semantics and Compilation;
- Formal Languages Theory and Automata;
- Decision Procedures and Certified Algorithms;
- Concurrent Systems and Protocols;
- Operating Systems;
- Biology and Bio-CS.
Example -> demo

give the definitions of the objects one wants to model

**Inductive** nat : Type :=
  | O : nat
  | S : nat → nat.

**Fixpoint** plus (n m:nat) {struct n} : nat :=
  match n with
  | O ⇒ m
  | S p ⇒ S (p + m)
  end
where "n + m" := (plus n m) : nat_scope.

prove properties of these objects

**Lemma** plus_n_Sm : ∀ n m:nat, S (n + m) = n + S m.
**Proof.**
  intros n m.
  induction n;
  [simpl; trivial|
  simpl; rewrite IHn; trivial].
**Qed.**
Context

- using proof assistants to verify numerical algorithms
- formalization of mathematics in Coq (multivariate analysis)
Newton’s method

- find the root of a function $f$
- definition: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- Kantorovitch’s theorem gives sufficient conditions for the convergence of Newton’s method to the root of the function $f$
- it holds in the general case of a system of $p$ equations with $p$ variables
Proof Assistants

Coq

Verifying numerical algorithms

Formalizing mathematics
Kantorovitch’s theorem in the real case

Given the equation $f(x) = 0$, with $f : ]a, b[ \rightarrow \mathbb{R}$, $a, b \in \mathbb{R}$, $f(x) \in C^{(1)}(]a, b[)$ and $x^{(0)} \in ]a, b[$ so that $\overline{U}_\varepsilon(x^{(0)}) = \{|x - x^{(0)}| \leq \varepsilon\} \subset ]a, b[$.

If:

1. $f'(x^{(0)}) \neq 0$ and $\left| \frac{1}{f'(x^{(0)})} \right| \leq A_0$;

2. $\left| \frac{f(x^{(0)})}{f'(x^{(0))}} \right| \leq B_0 \leq \frac{\varepsilon}{2}$;

3. $\forall x, y \in [a, b], |f'(x) - f'(y)| \leq C|x - y|$

4. $2A_0B_0C \leq 1$.

Then, Newton’s method: $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$ converges and $x^* = \lim_{n \rightarrow \infty} x^{(n)}$ is the unique solution of the initial equation in the domain $\{|x^* - x^{(0)}| \leq 2B_0\}$. 
The problem with real numbers

- the real numbers are not representable on a computer
  - infinite set $\rightarrow$ finite set
  - several models: float, double, arbitrary precision, approximations using interval arithmetic etc.

- the floating point numbers are not suitable for proofs
  - they do not respect classic properties: associativity of the addition etc.
  - presence of concepts like underflow, overflow etc.
Possible solution

- do proofs on “classic” reals
- implement the algorithms on “machine” reals
- link the 2 representations in order to verify the algorithms
**The proofs**

- in Coq, the standard library \texttt{Reals}
- axiomatic definition, i.e. impose the expected mathematical properties: \((x + y) + z = x + (y + z)\) etc.

\[
\begin{align*}
\text{Variable} & \quad a \ b \ A0 \ B0 \ C \ X0 : R. \\
\text{Variable} & \quad f : R \to R. \\
\text{Hypothesis} & \quad \text{Hder} f : \forall x, \ a < x < b \to \text{derivable}\_\text{pt} f x. \\
\ldots & \\
(*\text{code the hypotheses of the theorem}*) & \\
\ldots & \\
\text{Fixpoint} & \quad \text{Xn} (f : R \to R) (f' : R \to R) (X0 : R) (n : \text{nat}) : R := \\
& \quad \begin{cases} \\
& \text{match} \ n \ \text{with} \\
& \quad |0 \Rightarrow X0 \\
& \quad |S \ n \Rightarrow \text{Xn} \ n - f(\text{Xn} \ n) / f'(\text{Xn} \ n) \\
& \end{cases} \\
\ldots & \\
\text{Theorem} & \quad \text{Kanto}\_\text{exist}: \\
& \exists xs : R, \ \text{conv} \ \text{Xn} \ xs \land f \ xs = 0.
\end{align*}
\]
The algorithms

use a model for “machine” reals

- e.g. reals with arbitrary precision can be modeled with infinite streams of digits
  \[ 0, d_1 d_2 d_3 \ldots \]

- encode Newton’s method on this type of reals

```coq
Fixpoint mXn (g: mR -> mR) (g':mR -> mR) (mX0: mR) (n: nat):mR:=
  match n with
  | O => mX0
  | S n => mXn n - g (mXn n) / g' (mXn n)
end.
```
• say that a “machine” real $x$ represents a certain “classical” real $r$

$$\text{represents } x \text{ } r$$

• do reasoning steps like

$$\text{represents } x_1 \text{ } r_1 \land \text{represents } x_2 \text{ } r_2 \rightarrow \text{represents } (x_1 \oplus x_2) (r_1 + r_2)$$

• ... in order to prove: $\forall f : R \rightarrow R, g : mR \rightarrow mR$

$$\text{represents } x_0 \text{ } r_0 \land (\forall x \text{ } r, \text{represents } x \text{ } r \rightarrow \text{represents } g(x) f(r))$$

$$\Rightarrow \forall n, \text{represents } (mXn \text{ } g \text{ } g' \text{ } x_0 \text{ } n) (Xn \text{ } f \text{ } f' \text{ } r_0 \text{ } n)$$

• and the root of $g$ represents the root of $f$
Proof Assistants

Coq

Verifying numerical algorithms

Formalizing mathematics
Kantorovitch’s theorem

Let \( f(x) = 0 \) be a system with \( p \) equations and \( p \) variables, with 
\[ f(x) \in C^{(2)}(\omega) \] 
and 
\[ \overline{U_\varepsilon}(x^{(0)}) = \{ \|x - x^{(0)}\| \leq \varepsilon \} \subset \omega. \]

If:

- the Jacobian matrix \( W(x) = \left[ \frac{\partial f_i}{\partial x_j} \right] \) for \( x = x^{(0)} \) has an inverse 
  \( \Gamma_0 = W^{-1} \) with \( \|\Gamma_0\| \leq A_0; \)
- \( \|\Gamma_0 f(x^{(0)})\| \leq B_0 \leq \frac{\varepsilon}{2}; \)
- \( \sum_{k=1}^{p} \left| \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k} \right| \leq C \) for \( i, j = 1, 2, \ldots, p \) and \( x \in \overline{U_\varepsilon}(x^{(0)}); \)
- \( 2pA_0B_0C \leq 1. \)

Then, Newton’s process: 
\[ x^{(n+1)} = x^{(n)} - W^{-1}(x^{(n)})f(x^{(n)}) \]
converges and \( x^* = \lim_{n \to \infty} x^{(n)} \) is the unique solution of the initial system in the domain \( \|x - x^{(0)}\| \leq 2B_0. \)
Organization of the multidimensional proof

- Axiomatic reals
- Finite types, indexed operations
- Vectors: operations, norm
- Sequences
- Functions
- Matrix
- Matrix norm
- Derivation
- Taylor’s formula
- Kantorovitch
Interesting references

for an introduction to proof assistants

- H. Barendregt and H. Geuvers, *Proof Assistants using Dependent Type Systems*
  at http://www.cs.ru.nl/herman/PUBS/HBKassistants.ps.gz

for details on the Coq proof system

- http://coq.inria.fr/
- Y. Bertot, P. Casteran, *Coq’Art: the Calculus of Inductive Constructions*

for details on formalization of numerical analysis in Coq

- M. Mayero, *Using Theorem Proving for Numerical Analysis*, at

for a description of the exact arithmetic library based on co-inductive streams

- N. Julien, *Certified exact real arithmetic using co-induction in arbitrary integer base* at http://hal.inria.fr/inria-00202744