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:

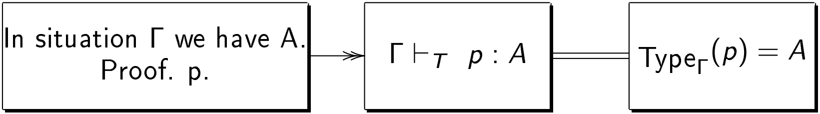
- based on set theory (Tarski - Grothendieck): Mizar
- based on type theory: HOL, Isabelle, Coq, ACL2, PVS, Agda, Lego, Nurpl, Minlog etc.

Why type theory?

a powerful formal system that captures

- *computation* (via the inclusion of functional programs written in typed λ-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



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

Coq

- *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. $\Longrightarrow$ 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 $\Longrightarrow$ 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, guardedeness...;
- 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 simultaneiously; $\Longrightarrow$ Research in Coq (3 researchers);
- dependent types;
- type theory $\Longrightarrow$ 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

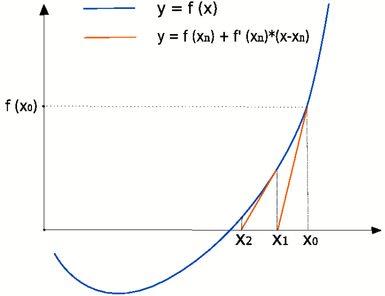
```
Lemma plus_n_Sm : ∀ n m:nat, S (n + m) = n + S m.
Proof.
 intros n m.
 induction n;
   [simpl; trivial|
   simpl; rewrite IHn; trivial].
Qed.
```

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 $f :]a, b[\to \mathbb{R}$, 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 $|\frac{1}{f'(x^{(0)})}| \leq A_0$;
2. $|\frac{f(x^{(0)})}{f'(x^{(0)})}| \leq B_0 \leq \frac{\varepsilon}{2}$;
3. $\forall x, y \in [a, b], |f'(x) - f'(y)| \leq C|x - y|$
4. $2A_0 B_0 C \leq 1$.

Then, Newton's method: $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$ converges and $x^* = \lim_{n \to \infty} x^{(n)}$ 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 $\longrightarrow$ finite set
  - several models: float, double, arbitrary precision, apriximations 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 `Reals`
- axiomatic definition, i.e. impose the expected mathematical proprieties: $(x+y)+z = x+(y+z)$ etc.

```
Variable a b A0 B0 C X0: R.
Variable f: R → R.
Hypothesis Hder_f: ∀ x, a < x < b → derivable_pt f x.
...
(*code the hypotheses of the theorem*)
...
Fixpoint Xn (f: R → R) (f': R → R)(X0: R)(n: nat):R:=
 match n with
  |O ⇒ X0
  |S n ⇒ Xn n - f(Xn n) / f'(Xn n)
 end.
...
Theorem Kanto_exist:
∃ xs: R, conv Xn  xs ∧ f xs = 0.
```

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

```
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.
```

The link

- say that a "machine" real x represents a certain "classical" real r

$$represents\ x\ r$$

- do reasoning steps like

$$represents\ x_1\ r_1 \wedge represents\ x_2\ r_2\ \longrightarrow represents\ (x_1 \oplus x_2)\ (r_1 + r_2)$$

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

$$represents\ x0\ r0 \wedge (\forall x\ r, represents\ x\ r \rightarrow represents\ g(x)\ f(r))$$

$$\Rightarrow \forall n, represents\ (mXn\ g\ g'\ x0\ n)\ (Xn\ f\ f'\ r0\ n)$$

- and the root of g represents the root of f

Proof Assistants

Coq

Verifying numerical algorithms

Formalizing mathematics

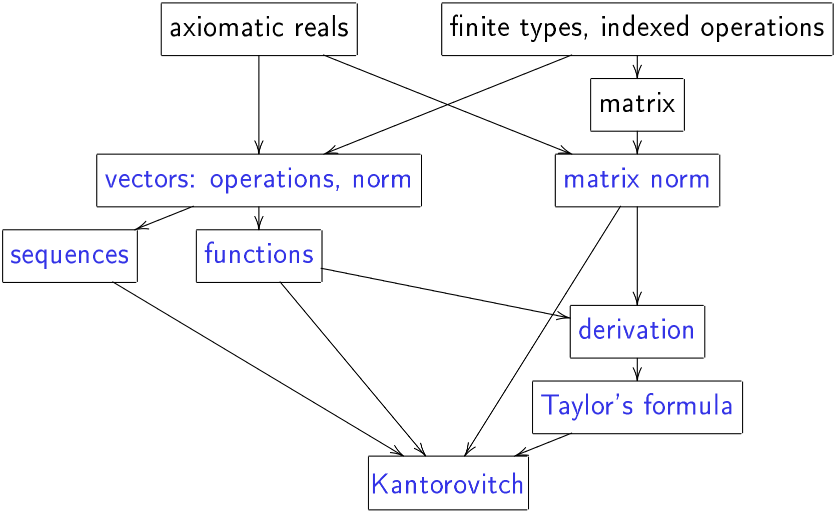
Kantorovitch's theorem

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 $W(x) = [\frac{\partial f_i}{\partial x_j}]$ for $x = x^{(0)}$ has an inverse $\Gamma_0 = W^{-1}$ with $\|\Gamma_0\| \leq A_0$;
- $\|\Gamma_0 f(x^{(0)})\| \leq B_0 \leq \frac{\varepsilon}{2}$;
- $\sum_{k=1}^{p} |\frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}| \leq C$ for $i, j = 1, 2, ..., p$ and $x \in \overline{U_\varepsilon}(x^{(0)})$;
- $2pA_0 B_0 C \leq 1$.

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



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 ftp://ftp.inria.fr/INRIA/LogiCal/Micaela.Mayero/papers/odyssee.ps.gz

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