Formalizing Constructive Mathematics in Type Theory

ZIC-Colloquim, T.U. Eindhoven

16 March 2004

Luís Cruz-Filipe

NIII, U. Nijmegen, Netherlands

CLC, Lisbon, Portugal

From 1.9.2004 *the University of Nijmegen will be called Radboud University of Nijmegen*

Contents

Contents

Formalizing Mathematics: What, Why, How?

What?

A computer representation of mathematical objects

Why?

Correctness

Applications

Presentation & Exchange

How?

In Coq

Why Coq?

- Type theory with inductive types
- Constructive logic
- Proof objects (de Bruijn criterion)
- Widely-used system
- Possible applications

Contents

The Constructive Coq Repository @ Nijmegen

What?

A library of constructive mathematics formalized in Coq

Where?

Repository: University of Nijmegen

Users: Nijmegen, France, (some day) all over the world

Why?

Formalize mathematics in a uniform way

The FTA-project

Objectives: Show it is possible to formalize non-trivial piece of mathematics.

Goal: Formalize the FTA in a modular and reusable way.

Period: 1999–2001

Achievements: Algebraic Hierarchy with axiomatic real numbers; specialized automation strategies; model of $\mathbb{R}$.

People: H. Barendregt, H. Geuvers, M. Niqui, R. Pollack, F. Wiedijk, J. Zwanenburg

Real Analysis & C-CoRN

Objectives: Reuse, test and extend the FTA-library.

Goal: Formalize 1^{st} year real calculus and identify where the main problems are.

Period: Sep/2001–Dec/2002

Achievements: Partial functions, differential & integral calculus, specialized tactics, library of transcendental functions

People: L. Cruz-Filipe

C-CoRN & CoRN

Goal: Expand in new directions.

- Program extraction (L. Cruz-Filipe, B. Spitters, Oct/2002–Dec/2003)

- Metric spaces (I. Loeb, Mar–Jun/2003)

- Group theory (H. Barendregt, D. Synek, Jun/2003–)

- Complex exponential (S. Hinderer, Jun–Jul/2003)

- Models and counter-examples (I. Loeb, Dec/2003–)

- Automation (L. Cruz-Filipe, D. Hendriks, F. Wiedijk)

- Maintenance (L. Cruz-Filipe, L. Mamane)

- Theoretical aspects (H. Barendregt, L. Cruz-Filipe, H. Geuvers, B. Spitters, F. Wiedijk)

Examples from the library

algebra	:	$\forall_{f:R[\mathbb{C}]}.(\text{nonConstant } f) \Rightarrow \exists_{z:\mathbb{C}}.f(z) = 0$
trigonometry	:	$\forall_{x:\mathbb{R}}.\cos(x)^2 + \sin(x)^2 = 1$
complex numbers	:	$e^{i\pi} + 1 = 0$

Methodology

- Aim at generality
- Reusability and extendability
- Constructive reasoning
- Two-sorted logic
- Visibility
- Colaboration with other projects (Coq, MoWGLI)
- Meta-analysis

Contents

Partial Functions with TCC's

Motivation: any partial function $F : S \not\to S$ is associated with a *domain* of definition D_F

$\leadsto$ $F(x)$ is defined whenever $D_F(x)$

$$\frac{x : S \quad F : S \not\to S \quad H : D_F(x)}{Fx : S}$$

$\leadsto$ used in e.g. NuPRL, PVS

$\leadsto$ undecidable type checking

Treatment of subsetoids

If $P : S \to \mathsf{Prop}$, then $\{S|P\}$ is the subsetoid of elements of S satisfying P

$\rightsquigarrow$ encoded in type theory:

$$\frac{x : \{S|P\}}{x : S} \qquad\qquad \frac{x : S \quad H : P(x)}{x : \{S|P\}}$$

$\rightsquigarrow$ second rule again yields undecidability

Examples of Partial Functions

$$
\begin{array}{rcl}
\text{Expon} & := & \lambda x : \mathbb{R}. \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
\text{H} & : & \forall x : \mathbb{R}. D_{\text{Expon}}(x) \\
\text{Exp} & := & \lambda x : \mathbb{R}. \text{Expon}(x, H(x))
\end{array}
$$

$$
\begin{array}{rcl}
\text{Log} & := & \lambda x : \mathbb{R}. \int_1^x \frac{1}{t} dt \\
D_{\text{Log}}(x) & \to & x > 0
\end{array}
$$

$\rightsquigarrow$ trigonometric functions defined in a similar way

Contents

Equational Reasoning in the Algebraic Hierarchy

Motivation: mathematical proofs often require manipulating equalities involving complex expressions.

Three main tactics:

Algebra

Context-sensitive, easy-to-extend search tactic (“`Auto with ...`”)

Rational

Reflection-based tactic for fields

Step

Allows the user to “replace equals by equals” on (some) goals

Search tactics: Algebra

+ Uses hypotheses from the context

+ Can be extended any time a new lemma is proved

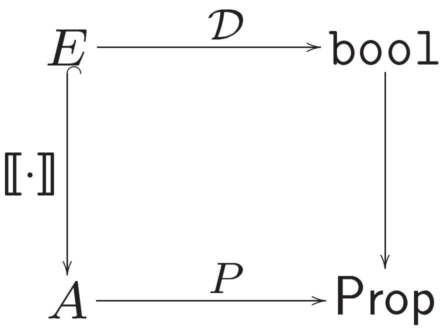
+ Quick and efficient for simple goals, e.g. $x = a \to x+y = a+y$

– Limited usage

– Can take a long time to fail

– Not modular(!)

Reflection tactics: Rational



such that $(\mathcal{D}(e) = \top) \to P([\![e]\!])$

In our case: E consists of (formal) rational functions, $\mathcal{N} : E \to E$ rewrites each rational function to a normal form, and the correctness lemma states that

$$(\mathcal{N}(a - b) = 0/e) \to [\![a]\!] = [\![b]\!].$$

Rational: Hierarchical version

$\rightsquigarrow$ From the definition of $\mathcal{N}$ one can immediately see that the same implementation yields tactics for rings and (abelian) groups.

$\rightsquigarrow$ It is also easy to treat arbitrary (partial) function symbols.

but. . .

$\rightsquigarrow$ Completeness is lost if the following two axioms are coupled:

$$(\mathbf{F})\forall x.(x \neq 0 \rightarrow x \times \frac{1}{x} = 1)$$

$$(\mathbf{Set_4})\forall f.\forall x, y.(x = y \rightarrow f(x) = f(y))$$

Properties of Rational

+ Proved complete
+ Follows the Algebraic Hierarchy
+ Linear length of proof terms
+ No bound on the complexity of the proof (. . .)
– Too expensive for simple goals
– Disregards context
– Not extensible

The Step tactic

If $a = c$, to go from $a < b$ to $c < b$ one needs more than just the ability to prove equalities.

Step: Collects several lemmas of the forms

$$a\mathcal{R}b \to a = c \to c\mathcal{R}b$$

$$a\mathcal{R}b \to b = c \to a\mathcal{R}c$$

and chooses the one to use according to the goal.

Examples

Other tactics in C-CoRN

Contin Proves that a given function is continuous on an interval

Deriv Partially solves $f' = g$ on a given interval

SetoidRewrite Replaces a with b everywhere in the goal, assuming that $a = b$

Contents

Why?

- Large constructive library
- Coq has extraction mechanism
- It doesn't work

(joint work with Bas Spitters and Pierre Letouzey)

Extraction

- BHK-interpretation: connectives

- Kleene's realizability: a more formal approach

- Curry–Howard isomorphism: proofs $\longleftrightarrow$ programs

- In practice: algorithm vs. properties; types as "markers"

FTA-logic

- No elimination of Prop terms over $\mathbf{Set}$ $\rightsquigarrow$ no function definition by cases

- All logic in $\mathbf{Set}$

- Extracted program too big

A solution?

Identify *computationally meaningful* propositions; put everything else in Prop.

$\rightsquigarrow$ most proof terms can be put back in Prop

$\rightsquigarrow$ significant amount of “dead code” is eliminated

Connectives

$$\neg \quad : \quad s \to \mathsf{Prop}$$

$$\to \quad : \quad s_1 \to s_2 \to s_2$$

$$\vee \quad : \quad s_1 \to s_2 \to \mathbf{Set}$$

$$\wedge \quad : \quad s_1 \to s_2 \to \begin{cases} \mathsf{Prop} & s_1 = s_2 = \mathsf{Prop} \\ \mathbf{Set} & \text{otherwise} \end{cases}$$

$$\forall \quad : \quad \Pi(A : s_1).(A \to s_2) \to s_2$$

$$\exists \quad : \quad \Pi(A : \mathbf{Set}).(A \to s) \to \mathbf{Set}$$

Results

- FTA: extracts, compiles, runs. . . but does not terminate

- Rational numbers: everything is (almost) instantaneous

- Somewhere in between: e, π and $\sqrt{2}$

Computing e

$$e \stackrel{\text{def}}{=} \sum_{n=0}^{+\infty} \frac{1}{k!}$$

$\rightsquigarrow$ each term is a rational (constant sequence)

$\rightsquigarrow$ but much is going on...

Immediate Problems. . .

- Unary natural numbers
- A direct proof of $k! \neq 0$ requires computing $k!$ in unary notation

. . . & Solutions

- Directly inject $\mathbb{Z}^+$ into $\mathbb{R}$
- Prove $k! \neq 0$ by induction on k

Still better

Optimize performance by working directly in the model:

- More efficient definition of factorial

- Simpler proofs and smaller proof terms

$\rightsquigarrow$ 100^{th} approximation in 77 seconds (with 157 correct digits)

The next step: $\sqrt{2}$

Different constructive formulations of the IVT...

- for total functions
- for partial functions
- for monotone functions
- for locally non-constant functions
- for polynomials

. . . and different extracted programs:

- new $\sqrt{2}$ now yields first approximation after just 6 seconds (instead of 52 hours)

- complexity is still exponential

- key lemma (for increasing version of IVT)

  $a < b \Rightarrow f(a) < f(b)$, where f is the function being iterated

Any future in this?

- The more abstract the formalization, the less efficient the extracted program

- Obtaining a *working* program is far from straightforward

- Small, carefully thought, modifications in the formalization can make huge differences in the extracted program

- Future improvements in Coq may also make huge differences. . .

Contents

Conclusions

- Large and growing library of formalized mathematics

- Satisfactory (though not ideal) treatment of partiality

- Large variety of domain-specific tactics

- Programs from proofs? Maybe some day...

PhD defense

on Tuesday, June 15 at 15.30

in the Aula of the U. Nijmegen