Types Summer School
Gothenburg Sweden August 2005

Lecture 5: FTA, the Fundamental Theorem of ALgebra
C-CoRN, The Constructive Coq Repository @ Nijmegen

Herman Geuvers, Luis Cruz-Filipe, Freek Wiedijk, Milad Niqui,
Jan Zwanenburg,
Randy Pollack, Iris Loeb, Bas Spitters, Sebastien Hinderer,
Henk Barendregt, Dan Synek
Radboud University Nijmegen, NL

What, Where, Why

- What: A coherent library of formalized mathematics
- Where: @ Nijmegen (NL), but possibly users and contributors from all over the world.
- Why: formalize mathematics in a uniform way.

What? Content

- Algebraic Hierarchy: monoids, rings, (ordered) fields, ...
- Tactics, esp. for equational reasoning
- Real number structures: axiomatically as complete Archimedean ordered fields.
- Model of $\mathbb{R}$ + proof that two real number structures are isomorphic + alternative axioms
- Generic results about $\mathbb{R}$ and $\mathbb{R}$-valued functions
- (Original) FTA-library: definition of $\mathbb{C}$ and proof of FTA
- Real analysis following Bishop: Continuity, differentiability and integrability, Rolle's Theorem, Taylor's Theorem, FTC. The exponential and trigonometric functions, logarithms and inverse trigonometric functions.

The sizes of the C-CoRN library:

Description	Size (Kb)	% of total
Algebraic Hierarchy (incl. tactics)	533	26.4
Real Numbers (incl. Models)	470	23.3
FTA (incl. Complex Numbers)	175	8.7
Real Analysis (incl. Transc. Fns.)	842	41.6
Total	2020	100

Why? Aims

- Not one (isolated) big fancy theorem, but create a library:
  "Mexican hat"

Sets and Basics	41 kb
Algebra (upto Ordered Fields)	165 kb
Reals	52 kb
Polynomials	113 kb
Real-valued functions / Basic Analysis	30 kb
Complex numbers	98 kb
FTA proof	70 kb
Construction of $\mathbb{R}$ (Niqui)	309 kb
Rational Tactic	49 kb

Aims ctd.

- Make interaction between different fields of mathematics possible.
- Reusable by others: take care of documentation, presentation, notation, searching
- Constructive(?)
  Finer analysis of mathematics, esp. analysis: reals are (potentially) infinite objects; computational content.
- Formalizing math. on a computer is fun, but also has benefits:
  - Correctness guaranteed.
  - Exchange of 'meaningful' mathematics.
  - Finding mathematical results.

Aims ctd.

- Investigate the current limitations.
- Try to manage this project. Three sequential/parallel phases:

Mathematical proof	LaTeX document (lots of details)
Theory development	Coq file (just defs and statements of lemmas)
Proof development	Coq file (proofs filled in)

Try to keep these phases consistent!

Problems ?

- Idiosyncrasies of 'the' Proof Assistant.
- Verbosity of formalized mathematics.
- Access to the formalized mathematics.

Methodology
Work in a systematic way (CVS):

- Documentation: what has been formalized; notations; definitions; tactics.
- Structuring: Group Lemmas and Def's according to mathematical content; Name Lemmas and Def's consistently.
- Axiomatic Approach: C-CoRN aims at generality.
- Automation: Develop tactics for specific fields of mathematics

A brief look into C-CoRN

- (Constructive) Setoids
- Algebraic Hierarchy
- Partial Functions
- $\mathbb{R}$
- FTA proof
- Automation via Reflection

Setoids
How to represent the notion of set?
Note: A set is not just a type, because
$M : A$ is decidable whereas $t \in X$ is undecidable
A setoid is a pair $[A, =]$ with

- A : Set,
- $= : A{\to}(A{\to}\mathsf{Prop})$ an equivalence relation over A

A setoid function is an $f{:}A{\to}B$ such that

$$\forall x, y{:}A.(x =_A y){\to}(f\ x) =_B (g\ y).$$

Here: Constructive Setoids
Apartness $\#$ as basic:

$$\begin{aligned} x = y &\leftrightarrow \neg(x \# y) \\ x \# y &\to (x \# z) \vee (y \# z) \\ \neg(x \# x)& \\ x \# y &\to y \# x \end{aligned}$$

A constructive setoid function is an $f{:}A{\to}B$ such that

$$\forall x, y{:}A.(f\ x) \#_B (g\ y){\to}(x \#_A y).$$

Strong extensionality

The algebraic hierarchy

- We deal with real numbers, complex numbers, polynomials, ...
- Many of the properties we use are generic and algebraic.
- To be able to reuse results and notation we have defined a hierarchy of algebraic structures.
- Basic level: constructive setoids.
- Next level: semi-groups, $\langle S, +\rangle$, with S a setoid and $+$ an associative binary operation on S.

Structures and Coercions

```
Record CMonoid : Type :=
  { m_crr    :> CSemi_grp;
    m_proof :  (Commutative m_crr (sg_op m_crr))
        /\ (IsUnit m_crr (sg_unit m_crr) (sg_op m_crr))
  }.
```

- A monoid is now a tuple $\langle\langle\langle S, =_S, r\rangle, a, f, p\rangle, q\rangle$
  If `M : Monoid`, the carrier of `M` is `(crr(sg_crr(m_crr M)))`
  Nasty !!
  $\Rightarrow$ We want to use the structure `M` as synonym for the carrier set `(crr(sg_crr(m_crr M)))`.
  $\Rightarrow$ The maps `crr`, `sg_crr`, `m_crr` should be left implicit.
- The notation `m_crr :> Semi_grp` declares the coercion `m_crr : Monoid >-> Semi_grp`.

Inheritance via Coercions

We have the following coercions.

```
OrdField >-> Field >-> Ring >-> Group
             Group >-> Monoid >-> Semi_grp >-> Setoid
```

- All properties of groups are inherited by rings, fields, etc.
- Also notation is inherited:

  `x[+]y`

  denotes the addition of `x` and `y` for `x,y:G` from any semi-group (or monoid, group, ring,...) `G`.
- The coercions must form a tree, so there is no real multiple inheritance:
  E.g. it is not possible to define rings in such a way that it inherits both from its additive group and its multiplicative monoid.

Partiality: Proof terms inside objects

- The 'subtype' $\{t : A \mid (P\ t)\}$ is defined as the type of pairs $\langle t, p\rangle$ where $t : A$ and $p : (P\ t)$.
  Notation: $\Sigma x{:}A.P\ x$
- A partial function is a function on a subtype
  E.g. $(-)^{-1} : \Sigma x{:}\mathbb{R}.x \neq 0 \to \mathbb{R}$.
  If $x : \mathbb{R}$ and $p : x \neq 0$, then $\frac{1}{\langle x,p\rangle} : \mathbb{R}$.
- A partialfunction must be proof-irrelevant, i.e.
  if $p : t \neq 0$ and $q : t \neq 0$, then $\frac{1}{\langle t,p\rangle} = \frac{1}{\langle t,q\rangle}$.
- For practical (Coq) purposes we "Curry" partial functions and take
  $(-)^{-1} : \Pi x{:}\mathbb{R}.(x \neq 0) \to \mathbb{R}$.

The Real Numbers in Coq:

- Axiomatic: a 'Real Number Structure' is a
  Cauchy-complete Archimedean ordered field.
- Prove FTA 'for all real numbers structures'.
- Construct a model to show that real number structures exist. (Cauchy sequences over an Arch. ordered field, say $\mathbb{Q}$)
- Prove that any two real number structures are isomorphic.

Axioms for Real Numbers:

- Cauchy sequences over Field F:
  g : nat $\to F$ is Cauchy if
  $$\forall \varepsilon{:}F_{>0}.\exists N{:}\mathbb{N}.\forall m \geq N(|g_m - g_N| < \varepsilon)$$
- All Cauchy sequences have a limit:
  $$\mathsf{SeqLim} : (\Sigma g{:}\mathsf{nat}{\to}F.\mathsf{Cauchy}\, g) \to F$$
  $$\mathsf{CauchyProp} : \forall g{:}\mathsf{nat}{\to}F.(\mathsf{Cauchy}\, g) \to \forall \varepsilon{:}F_{>0}.\exists N{:}\mathbb{N}.\forall m \geq N.(|g_m - (\mathsf{SeqLim}\, g)| < \varepsilon)$$
- Axiom of Archimedes: (there are no non-standard elements)
  $$\forall x{:}F.\exists n{:}\mathbb{N}(n > x)$$

NB: The axiom of Archimedes proves that 'ε-Cauchy sequences' and '$\frac{1}{k}$-Cauchy sequences' coincide (similar for limits)

Consequences of the Axiomatic approach:

- We don't construct $\mathbb{R}$ out of $\mathbb{Q}$, so we don't have $\mathbb{Q} \subset \mathbb{R}$ on with $=$ decidable on $\mathbb{Q}$.
- We did not want to 'define' $\mathbb{Q} \subset \mathbb{R}$.
- Instead: modify the proof by introducing fuzziness:
  Instead of having to decide
  $$x < y \vee x = y \vee x > y,$$
  all we need to establish is whether (for given $\varepsilon > 0$)
  $$x < y + \varepsilon \vee x > y - \varepsilon$$
  which we may write as
  $$x \leq_\varepsilon y \vee x \geq_\varepsilon y$$
  This is decidable, due to the cotransitivity of the order relation:
  $$x < y \Rightarrow x < z \vee z < y$$

Intermezzo Program Extraction

The logic of Coq (and most type theories) is constructive. This implies that

if $\vdash \forall x{:}A \exists y{:}B.R\, x\, y$, then there is a term f such that $\vdash \forall x{:}A.R\, x\, (f\, x)$.

Application: From a proof term of $\forall x \in \mathsf{nat}.\exists y \in \mathsf{nat}.x{+}x \leq y$ one can extract

- a term (Coq-program) f : nat$\to$nat,
- a proof of $\forall x{:}\mathsf{nat}.x + x \leq f\, x$ (correctness of f)

Strengthening

if $\vdash \forall x{:}A.P\, x \vee \neg P\, x$ and $\vdash \forall x{:}A.P\, x \to \exists y{:}B.R\, x\, y$, then there is a term f such that $\vdash \forall x{:}A.P\, x \to R\, x\, (f\, x)$.

Example

$$\forall l{:}\mathsf{list}.l \neq \mathtt{nil} \to \exists n{:}\mathsf{nat}.n \leq l \wedge n \in l$$

Pros/Cons of the Axiomatic approach:

Pros:

- "Plug-in" arbitrary (your own pet) model to extract algorithm.
- Work abstractly: reuse

Cons (?):

- Choice of axioms? Don't try to be minimal! E.g.maximum function should be added.
- Can we get "good" algorithms when we work abstractly?

FTA: The classical FTA proof

Suppose $|f(z)|$ is minimal with $|f(z)| \neq 0$.
We construct a z_0 with $|f(z_0)| < |f(z)|$.
We may assume that the minimum is reached for $z = 0$.

$$f(x) = a_0 + a_k x^k + O(x^{k+1})$$

with a_k the first coefficient that's not 0.
Now take

$$z_0 := \varepsilon \sqrt[k]{-\frac{a_0}{a_k}}$$

with $\varepsilon \in \mathbb{R}_{>0}$.
If ε is small enough, the part $O\left(z_0^{k+1}\right)$ will be negligible and we get a $z_0 \neq 0$ for which

$$|f(z_0)| = a_0 + a_k \left(\varepsilon \sqrt[k]{-\frac{a_0}{a_k}}\right)^k = a_0(1 - \varepsilon^k) < |f(0)|$$

The constructive FTA proof

Define an algorithm
Given $z \in \mathbb{C}$, construct a sequence $z, z_0, z_1, \ldots$ going to the root.
Problem: in the definition

$$z_0 := \varepsilon \sqrt[k]{-\frac{a_0}{a_k}}$$

- ε must be small enough to neglect $O\left(z_0^{k+1}\right)$
- ε must be large enough to reach the root.

Solution (Kneser): write

$$f(x) = a_0 + a_k x^k + \text{other terms}$$

and find k and z_0 such that $|a_k||z_0|^k$ is big enough w.r.t. the other terms and small enough compared to $|a_0|$.

Automation via Computation

Poincaré Principle (Barendregt)

> "An equality involving a computation does not require a proof"

In type theory: if $t = q$ by evaluation (computing an algorithm), then this is a trivial equality, proved by reflexivity.
This is made precise by the conversion rule:

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : B}\ A =_{\beta\iota\delta} B$$

Can one actually use the programming power of Type Theory when formalizing mathematics?

Yes. For automation: replacing a proof obligation by a computation

Reflection Suppose

- We have a class of problems with a syntactic encoding as a data type, say via the type Problem.
  Example: equalities between expressions over a group
  Then the syntactic encoding is

```
Inductive E : Set :=
    evar  : nat -> E
  | eone  : E
  | eop   : E -> E -> E
  | einv  : E -> E
```

- We have a decoding function $[\![-]\!]$: Problem $\to$ Prop
- We have a decision function Dec : Problem $\to \{0,1\}$
- We can prove Ok : $\forall p$:Problem((Dec$(p) = 1) \to [\![p]\!]$)

To verify P (from the class of problems):

- Find a p : Problem such that $[\![p]\!] = P$.
- Then Dec(p) yields either 1 or 0
- If Dec$(p) = 1$, then we have a proof of P (using Ok)
- If Dec$(p) = 0$, we obtain no information about P (it 'fails')

Note: if Dec is complete:

$$\forall p\text{:Problem}((\text{Dec}(p) = 1) \leftrightarrow [\![p]\!])$$

then Dec$(p) = 0$ yields a proof of $\neg P$.
This can be made into a tactic, e.g. `Rational`, that proves equalities between rational expressions.

Related Work:

- Mizar largest library of formalized math., MML (Trybulec)
- HOL-light (Harrisson)
- Isabelle (Fleuriot, non-standard reals)
- Nuprl (Howe, constructive á la Bishop)
- Classical Reals in Coq (Mayero)
- Minlog (Schwichtenberg)
- FOC (Hardin, Rioboo)

Some Conclusions:

- Real mathematics, involving algebra and analysis can be formalised completely within a theorem prover (Coq).
- Setting up a basic library and some good proof automation procedures is a large part of the work.
- Library can be reused: Luis Cruz-Filipe proved FTC (and more).
- Extracting algorithms (e.g. for FTA) requires a further analysis of the proof (Luis Cruz-Filipe, Bas Spitters).
- In the end, the computational behaviour of algorithms should depend mainly on the representation of the reals.