Solving Ordinary Differential Equations in Coq

Evgeny Makarov
Joint work with Bas Spitters

Radboud University
Nijmegen

19 November 2012

Outline

Outline

Type Classes

Type classes are parametric record types.
Coq searches for terms of these types automatically during unification.

Two most often type class uses:

(1) Operational type classes
(2) Type classes representing mathematical structures

B. Spitters, E. van der Weegen, *Type Classes for Mathematics in Type Theory*, 2011.

Operational Type Classes

```
Coq < Class Plus A := plus: A -> A -> A.

Coq < Print plus.
plus = λ (A : CProp) (p : Plus A), p
  : forall A : Type, Plus A -> A -> A -> A
Arguments A, p are implicit and maximally inserted

Coq < Instance nat_plus: Plus nat := Peano.plus.

Coq < Variables x y : nat.
```

plus x y = @plus nat nat_plus x y ↠ nat_plus x y →
Peano.plus x y

```
Coq < Infix "+" := plus : mc_scope.
```

Type Classes for Mathematical Structures

```
Class AppRationals AQ {e plus mult zero one inv} `{Apart AQ}
    `{Le AQ} `{Lt AQ}
     {AQtoQ : Cast AQ Q_as_MetricSpace}
    `{!AppInverse AQtoQ} {ZtoAQ : Cast Z AQ}
    `{!AppDiv AQ} `{!AppApprox AQ}
    `{!Abs AQ} `{!Pow AQ N} `{!ShiftL AQ Z}
    `{∀ x y : AQ, Decision (x = y)}
    `{∀ x y : AQ, Decision (x ≤ y)} : Prop := {
  aq_ring :> @Ring AQ e plus mult zero one inv ;
  aq_trivial_apart :> TrivialApart AQ ;
  aq_order_embed :> OrderEmbedding AQtoQ ;
  aq_strict_order_embed :> StrictOrderEmbedding AQtoQ ;
  aq_ring_morphism :> SemiRing_Morphism AQtoQ ;
  aq_dense_embedding :> DenseEmbedding AQtoQ ;
  aq_div : ∀ x y k, ball (2 ^ k) ('app_div x y k) ('x / 'y) ;
  aq_compress : ∀ x k, ball (2 ^ k) ('app_approx x k) ('x) ;
  aq_shift :> ShiftLSpec AQ Z (≪) ;
  aq_nat_pow :> NatPowSpec AQ N (^) ;
  aq_ints_mor :> SemiRing_Morphism ZtoAQ
}.
```

B. Spitters, R. Krebbers, *Type classes for efficient exact real arithmetic in Coq*

Instances of Approximate Rationals

```
Record Dyadic Z := dyadic { mant: Z; expo: Z }.
```

Represents $\mathtt{mant} \cdot 2^{\mathtt{expo}}$

```
Instance dy_mult: Mult Dyadic :=
  λ x y, dyadic (mant x * mant y) (expo x + expo y).

Instance : AppRationals (Dyadic bigZ).

Instance : AppRationals bigQ.

Instance : AppRationals Q.
```

Outline

Metric Spaces

Let (X, d) where $d : X \to X \to \mathbb{R}$ be a metric space.

Let $Brxy$ denote $d(x, y) \leq r$.

A function $f : \mathbb{Q}^+ \to X$ is called regular if
$\forall \varepsilon_1\ \varepsilon_2 : \mathbb{Q}^+,\ B(\varepsilon_1 + \varepsilon_2)(f\varepsilon_1)(f\varepsilon_2)$.

The completion $\mathfrak{C}X$ of X is the set of regular functions.

Let X and Y be metric spaces. A function $f : X \to Y$ is called uniformly continuous with modulus μ if
$\forall \varepsilon : \mathbb{Q}^+\ \forall x_1\ x_2 : X,\ B(\mu\varepsilon)x_1x_2 \to B\varepsilon(fx_1)(fx_2)$.

If $x_1, x_2 : \mathfrak{C}X$, let $B_{\mathfrak{C}X}\varepsilon x_1x_2 := \forall \varepsilon_1\varepsilon_2 : \mathbb{Q}^+,\ B_X(\varepsilon_1 + \varepsilon + \varepsilon_2)(x_1\varepsilon_1)(x_2\varepsilon_2$

Metric spaces with uniformly continuous functions form a category.

Completion forms a monad in the category of *prelength spaces* and uniformly continuous functions.

R. O'Connor, extending work by E. Bishop.

Completion as a Monad

unit : $X \to \mathfrak{C}\, X := \lambda x \lambda \varepsilon,\ x$

join : $\mathfrak{C}\,\mathfrak{C}\, X \to \mathfrak{C}\, X := \lambda x \lambda \varepsilon,\ x(\varepsilon/2)(\varepsilon/2)$

map : $(X \to Y) \to (\mathfrak{C}\, X \to \mathfrak{C}\, Y) := \lambda f \lambda x,\ f \circ x \circ \mu_f$

bind : $(X \to \mathfrak{C}\, Y) \to (\mathfrak{C}\, X \to \mathfrak{C}\, Y) :=$ join $\circ$ map

Define functions $\mathbb{Q} \to \mathbb{Q}$; lift to $\mathfrak{C}\,\mathbb{Q} \to \mathfrak{C}\,\mathbb{Q}$.

Efficient Reals

In CoRN, `MetricSpace` is a regular `Record`, not a type class.

```
Coq < Check Complete.
Complete : MetricSpace -> MetricSpace

Coq < Check Q_as_MetricSpace.
Q_as_MetricSpace : MetricSpace

Coq < Check AQ_as_MetricSpace.
AQ_as_MetricSpace :
  ∀ (AQ : Type) ..., AppRationals AQ -> MetricSpace

Coq < Definition CR := Complete Q_as_MetricSpace.

Coq < Definition AR := Complete AQ_as_MetricSpace.
```

`AR` is an instance of `Zero`, `Plus`, `Le`, `Field`, `FullPseudoSemiRingOrder`, etc., from the MathClasses library.

Outline

Banach Fixpoint Theorem

Theorem

Let X be a complete metric space and let $f : X \to X$ be a contraction, i.e., $\forall x_1\, x_2 : X\ \forall r : \mathbb{Q},\ B r x_1 x_2 \to B(qr)(f x_1)(f x_2)$ for some $0 \leq q < 1$. Then f has a unique fixpoint.

Already implemented

Uses `MetricSpaceClass`, `CompleteMetricSpaceClass`, `IsUniformlyContinuous`, `IsContraction` type classes

Based on the unfinished work by E. van der Weegen and B. Spitters.

Picard-Lindelöf Theorem

Theorem

Consider the initial value problem

$$y'(x) = F(x, y(x)), y(x_0) = y_0 \tag{1}$$

Suppose that F is continuous in x and Lipschitz continuous in y, i.e., $\forall x_1\, x_2 : X\ \forall r : \mathbb{Q},\ B r x_1 x_2 \to B(Lr)(f x_1)(f x_2)$ for some $0 \leq L$. Then there exists an $\varepsilon > 0$ such that the problem has a unique solution on $[x_0 - \varepsilon, x_0 + \varepsilon]$.

(1) is equivalent to

$$y(x) = y(x_0) + \int_{x_0}^{x} F(t, y(t))\, dt \tag{2}$$

Picard-Lindelöf Theorem

$$y(x) = y(x_0) + \int_{x_0}^{x} F(t, y(t))\, dt \tag{2}$$

Define

$$\begin{aligned}(Tf)(x) &= y_0 + \int_{x_0}^{x} F(t, f(t))\, dt \\ f_0(x) &= y_0 \\ f_{n+1} &= T f_n\end{aligned}$$

Then for every $\Delta y > 0$ there exists a $\Delta x > 0$ such that T is a contraction on (locally) uniformly continuous functions that map $[x_0 - \Delta x, x_0 + \Delta x]$ to $[y_0 - \Delta y, y_0 + \Delta y]$. By the Banach fixpoint theorem, T has a fixpoint.

Initial implementation uses CR (inefficient), existing code

Integral

Following M. Bridger, *Real Analysis: A Constructive Approach*.

```
Class Integral (f: Q -> CR) :=
  integrate: forall (from: Q) (w: QnonNeg), CR.

Notation "∫" := integrate.

Class Integrable `{!Integral f}: Prop := {
  integral_additive:
    forall (a: Q) b c, ∫ f a b + ∫ f (a + b) c == ∫ f a (b + c);

  integral_bounded_prim: forall (from: Q) (width: Qpos) (mid: Q)
    (forall x, from <= x <= from + width -> ball r (f x) mid) ->
    ball (width * r) (∫ f from width) (width * mid);

  integral_wd :>
    Proper (Qeq ==> QnonNeg.eq ==> @st_eq CRasCSetoid) (∫ f)
}.
```

Earlier (abstract, but slower) implementation of integral by R. O'Connor and B. Spitters.

Complexity

Rectangle rule:

$$\left| \int_a^b f(x)\,dx - f(a)(b-a) \right| \le \frac{(b-a)^3}{24} M$$

where $|f''(x)| \le M$ for $a \le x \le b$.

Number of intervals to have the error $\le \varepsilon$: $\ge \sqrt{\dfrac{(b-a)^3 M}{24\varepsilon}}$

Simpson's rule:

$$\left| \int_a^b f(x)\,dx - \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) \right| \le \frac{(b-a)^5}{2880} M$$

where $|f^{(4)}(x)| \le M$ for $a \le x \le b$.

T. Coquand, B. Spitters. *A constructive proof of Simpson's Rule*, 2012

Number of intervals: $\ge \sqrt[4]{\dfrac{(b-a)^5 M}{2880\varepsilon}}$

The number of points grows exponentially with the number of significant digits.

Outline

Future Work

Change the development from CR to AR based on dyadic rationals.

Implement Simpson's integration and prove its error bounds.

Make sure unnecessary precision is not required, i.e., approximations to function values are calculated with optimal precision. E.g.:

$$(x + y)(\varepsilon) = x(\varepsilon/2) + y(\varepsilon/2)$$

Allow a more flexible distribution of ε when the computation cost of the arguments is different.