Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Jorge O. Acevedo-Acosta and Jose L. Echeverri-Jurado

EAFIT University
(Work in Progress)

Logic and Computation Seminar
EAFIT University
October 20, 2015

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Abstract

We present the formalization of the completeness axiom in the proof assistant Agda and additionally the use of ATPs to prove different properties of real numbers.

Constant, Relationships and Basic Functions

postulate

$$\mathbb{R} : Set$$

$$r_0 : \mathbb{R}$$

$$r_1 : \mathbb{R}$$

$$\_+\_ : \mathbb{R} \to \mathbb{R} \to \mathbb{R}$$

$$-\_ : \mathbb{R} \to \mathbb{R}$$

$$\_*\_ : \mathbb{R} \to \mathbb{R} \to \mathbb{R}$$

$$\_^{-1} : \mathbb{R} \to \mathbb{R}$$

$$\_>\_ : \mathbb{R} \to \mathbb{R} \to Set$$

Logic

- Equality[1]

$$\textit{data}\ \_\equiv\_ : \mathbb{R} \to \mathbb{R} \to \textit{Set}\ \textit{where}$$
$$\textit{refl} : \{x : \mathbb{R}\} \to x \equiv x$$

[1] Adapted from Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 23.

[2] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 19.

Logic

- Equality[1]

$$\textit{data}\ \_\equiv\_ : \mathbb{R} \to \mathbb{R} \to \textit{Set}\ \textit{where}$$
$$\textit{refl} : \{x : \mathbb{R}\} \to x \equiv x$$

- Injection

$$\mathbb{N}2\mathbb{R} : \mathbb{N} \to \mathbb{R}$$
$$\mathbb{N}2\mathbb{R}\ (\textit{zero}) = r_0$$
$$\mathbb{N}2\mathbb{R}\ (\textit{succ}\ n) = \mathbb{N}2\mathbb{R}\ n + r_1$$

[1] Adapted from Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 23.

[2] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 19.

Logic

- Equality[1]

$$\begin{aligned}&\mathit{data}\ \_\equiv\_ :\ \mathbb{R} \to \mathbb{R} \to \mathit{Set}\ \mathit{where}\\&\quad \mathit{refl} :\ \{x : \mathbb{R}\} \to x \equiv x\end{aligned}$$

- Injection

$$\begin{aligned}&\mathbb{N}2\mathbb{R}\ :\ \mathbb{N}\ \to\ \mathbb{R}\\&\mathbb{N}2\mathbb{R}\ \ (\mathit{zero})\ \ \ =\ r_0\\&\mathbb{N}2\mathbb{R}\ \ (\mathit{succ}\ n)\ =\ \mathbb{N}2\mathbb{R}\ n\ +\ r_1\end{aligned}$$

- Disjunction[2]

$$\begin{aligned}&\mathbf{data}\ \_\vee\_\ (A\ B\ :\ \mathit{Set})\ :\ \mathit{Set}\ \mathit{where}\\&\quad \mathit{inj}_1\ :\ A\ \to\ A\ \vee\ B\\&\quad \mathit{inj}_2\ :\ B\ \to\ A\ \vee\ B\end{aligned}$$

[1] Adapted from Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 23.

[2] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 19.

Logic

- Bottom[3]

  data $\bot$: *Set where*

  $\bot\text{-}elim : \{A : Set\} \rightarrow \bot \rightarrow A$

  $\bot\text{-}elim\ ()$

[3] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 20.

[4] Ana Bove and Peter Dybjer (2009). Dependent Types at Work. p. 22.

Logic

- Bottom[3]

  $\mathbf{data}\ \bot\ :\ \mathit{Set}\ \mathit{where}$

  $\bot\text{-}\mathit{elim}\ :\ \{A\ :\ \mathit{Set}\}\ \to\ \bot\ \to\ A$
  $\bot\text{-}\mathit{elim}\ ()$

- Negation[3]

  $\neg\_\ :\ \mathit{Set}\ \to\ \mathit{Set}$
  $\neg\ A\ =\ A\ \to \bot$

[3] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 20.

[4] Ana Bove and Peter Dybjer (2009). Dependent Types at Work. p. 22.

Logic

- Bottom[3]

  $\mathbf{data}\ \bot\ :\ \mathit{Set}\ \mathit{where}$

  $\bot\text{-}\mathit{elim}\ :\ \{A\ :\ \mathit{Set}\}\ \to\ \bot\ \to\ A$
  $\bot\text{-}\mathit{elim}\ (\,)$

- Negation[3]

  $\neg\_\ :\ \mathit{Set}\ \to\ \mathit{Set}$
  $\neg\ A\ =\ A\ \to \bot$

- Existential[4]

  $\mathbf{data}\ \exists_r\ (P\ :\ \mathbb{R}\ \to\ \mathit{Set})\ :\ \mathit{Set}\ \mathit{where}$
  $\mathit{exist}\ :\ (x\ :\ \mathbb{R})\ \to\ P\ x\ \to\ \exists_r\ P$

[3] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 20.

[4] Ana Bove and Peter Dybjer (2009). Dependent Types at Work. p. 22.

The Field Axioms in Agda

```
postulate
  +comm : (x y : ℝ)     → x + y       ≡ y + x
  +asso : (x y z : ℝ)   → x + y + z   ≡ x + (y + z)
  +neut : (x : ℝ)       → x + r₀      ≡ x
  +inve : (x : ℝ)       → x + (-x)    ≡ r₀
  *comm : (x y : ℝ)     → x * y       ≡ y * x
  *asso : (x y z : ℝ)   → x * y * z   ≡ x * (y * z)
  *neut : (x : ℝ)       → x * r₁      ≡ x
  *inve : (x : ℝ)       → ¬ (x ≡ r₀)  → x * (x ⁻¹) ≡ r₁
  1≢0 : ¬ (r₁ ≡ r₀)
  dist : (x y z : ℝ)    → x * (y + z) ≡ x * y + x * z

{-# ATP axiom *-comm *-asso *-neut *-inve #-}
```

The Order Axioms in Agda

```
postulate
    >asym : {x y : ℝ}   → x > y  → ¬ (y > x)
    >tran : {x y z : ℝ} → x > y  → y > z → x > z
    +cong : {x y z : ℝ} → x > y  → z + x > z + y
    *cong : {x y z : ℝ} → z > r₀ → x > y
                        → z * x > z * y

    trichotomy : (x y : ℝ) → (x > y) ∨ (x ≡ y) ∨ (x < y)

{-# ATP axiom >-asym >-trans >-+-cong-l >-*-cong-l #-}
```

The Archimedean Axiom in Agda[5]

```
postulate
  archimedean : (x : ℝ) → ∃₁ (λ n → ℕ2ℝ n > x)
```

[5] Adapted from Alberto Ciaffaglione and Pietro Di Gianantonio (2010). Types for Proofs and Programs, A tour with constructive real numbers, p. 43.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

The Completeness Axiom

"A nonempty set E of real numbers is said to be **bounded above** provided there is a real number b such that $x \leq b$ for all $x \in E$: the number b is called an **upper bound** for E. ... A set that is bounded above need not have a largest member. But the next axiom asserts that it does have a smallest upper bound.

The Completeness Axiom Let E be a nonempty set of real numbers that is bounded above. Then among the set of upper bounds for E there is a smallest, or least, upper bound."[6]

[6] H. L. Royden and P. M. Fitzpatrick (2010). Real Analysis. p. 9.

Completeness Axiom in Agda[7]

```
-- Definitions.
UpperBound : (ℝ → Set) → ℝ → Set
UpperBound E ub = (x : ℝ) → E x → x ≤ ub

Bound : (ℝ → Set) → Set
Bound E = ∃ᵣ (λ ub → UpperBound E ub)

Lub : (ℝ → Set) → ℝ → Set
Lub E sup =
(UpperBound E sup) ∧ ((ub : ℝ) →
                      UpperBound E ub → sup ≤ ub)

-- Completeness Axiom.
postulate
 completeness : (E : ℝ → Set) → Bound E →
                ∃ᵣ (λ x → E x) → ∃ᵣ (λ sup → Lub E sup)
```

[7] Adapted from Coq proof assistant (8.4pl4).
https://coq.inria.fr/distrib/current/stdlib/Coq.Reals.Raxioms.html

Completeness Axiom in Agda

- Example

$A : \mathbb{R} \to Set$
$A\ x = x \leq 2$

A is bounded above by 2. A maximum is equal to 2.
As A $= (-\infty,\ 2]$, then the set of upper bounds of A is $(2,\ +\infty)$.
sup A $= 2$, sup A $\in$ A.

Real Numbers Formalization: Axiom Completeness and Automatic Reasoning

Combining Agda with Automatic Theorem Provers[8]

- We use the TPTP language as the input language for the ATPs

[8] Andrés Sicard-Ramírez (2015) pp 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Real Numbers Formalization: Axiom Completeness and Automatic Reasoning

Combining Agda with Automatic Theorem Provers[8]

- We use the TPTP language as the input language for the ATPs
- In TPTP syntax: `fof(name, role, formula).`
  `role` $\in$ {`axiom, definition, hypothesis, conjecture`}

[8] Andrés Sicard-Ramírez (2015) pp 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Real Numbers Formalization: Axiom Completeness and Automatic Reasoning

Combining Agda with Automatic Theorem Provers[8]

- We use the TPTP language as the input language for the ATPs
- In TPTP syntax: `fof(name, role, formula).`
  `role` $\in$ {`axiom, definition, hypothesis, conjecture`}
- Using the ATP-pragma

```
{-# ATP axiom A #-}
{-# ATP axiom B #-} or
{-# ATP axiom A B #-}
```

[8] Andrés Sicard-Ramírez (2015) pp 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers[9]

- To automatically prove a formula A , we shall postulate it and add the ATP-pragma {`-# ATP prove A #-`}

[9] Andrés Sicard-Ramírez (2015) p 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers[9]

- To automatically prove a formula A , we shall postulate it and add the ATP-pragma {-# ATP prove A #-}
- For example to automatically prove the ≡-+-cong-r

```
postulate
  ≡-+-cong-r : {x y z : ℝ} → x ≡ y → x + z ≡ y + z

{-# ATP prove ≡-+-cong-r #-}

in an Agda file called PropertiesATP.agda
```

[9] Andrés Sicard-Ramírez (2015) p 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers[9]

- To automatically prove a formula A , we shall postulate it and add the ATP-pragma {-# ATP prove A #-}
- For example to automatically prove the ≡-+-cong-r

```
postulate
  ≡-+-cong-r : {x y z : ℝ} → x ≡ y → x + z ≡ y + z

{-# ATP prove ≡-+-cong-r #-}

in an Agda file called PropertiesATP.agda
```

- $ apia PropertiesATP.agda

```
Proving the conjecture
in /tmp/PropertiesATP/10-8744-comm.tptp
Vampire 0.6 (revision 903) proved the conjecture
```

[9] Andrés Sicard-Ramírez (2015) p 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers[10]

- If we want to only use one or more particular ATPs

```
$ apia --atp=e --atp=vampire PropertiesATP.agda
```

[10] Andrs Sicard-Ramrez (2015) pp 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs.
http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers[10]

- If we want to only use one or more particular ATPs

```
$ apia --atp=e --atp=vampire PropertiesATP.agda
```

- Using the ATP-pragma, we tell the ATPs to prove the theorems Examples

```
postulate
  ≡-*cong-r : {x y : ℝ}(z : ℝ) → x ≡ y →
              x * z ≡ y * z

{-# ATP prove ≡-*-cong-r #-}
```

[10] Andrs Sicard-Ramrez (2015) pp 91-114. Reasoning about Functional Programs by Combining Interactive and Automatic Proofs. http://www1.eafit.edu.co/asr/pubs/phd-thesis.pdf

Combining Agda with Automatic Theorem Provers

- postulate

```
≡-*-cong-l : {x y : ℝ}(z : ℝ) → x ≡ y →
             z * x ≡ z * y

{-# ATP prove ≡-*-cong-l #-}
```

Combining Agda with Automatic Theorem Provers

- postulate

```
≡-*-cong-l : {x y : ℝ}(z : ℝ) → x ≡ y →
             z * x ≡ z * y

{-# ATP prove ≡-*-cong-l #-}
```

- postulate

```
>-*-cong-r : {x y z : ℝ} → z > r₀ → x > y →
             x * z > y * z

{-# ATP prove >-*-cong-r #-}
```

Combining Agda with Automatic Theorem Provers

- postulate

```
≡-*-cong-l : {x y : ℝ}(z : ℝ) → x ≡ y →
             z * x ≡ z * y

{-# ATP prove ≡-*-cong-l #-}
```

- postulate

```
>-*-cong-r : {x y z : ℝ} → z > r₀ → x > y →
             x * z > y * z

{-# ATP prove >-*-cong-r #-}
```

- postulate

```
>-∧-*-cong-r : {x y z : ℝ} → (x > y) ∧ (z > r₀)
               → x * z > y * z

{-# ATP prove >-∧-*-cong-r >-*-cong-r #-}
```

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Future Work

- Combine interactive and automatic reasoning covering both proofs by induction and axiomatic.

  Prove $1^n = 1$

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Future Work

- Combine interactive and automatic reasoning covering both proofs by induction and axiomatic.

  Prove $1^n = 1$

- case base : $1^0 = 1$

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Future Work

- Combine interactive and automatic reasoning covering both proofs by induction and axiomatic.

  Prove $1^{n} = 1$
- case base : $1^{0} = 1$
- case (succ n) : $1^{\mathtt{succ}\, n} = 1$

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Future Work

- Combine interactive and automatic reasoning covering both proofs by induction and axiomatic.

  Prove $1^{n} = 1$
- case base : $1^{0} = 1$
- case (succ n) : $1^{succ\ n} = 1$
- ATP-pragma
  case base {-# ATP prove case-base #-}
  case (succ n) {-# ATP prove case-(succ n) #-}

Definitions

- The Rule of $\equiv -elimination$[11]

$$subst : (P : \mathbb{R} \to Set) \to \{x\ y : \mathbb{R}\} \to x \equiv y \to P\ x \to P\ y$$
$$subst\ P\ refl\ Px = Px$$

[11] Adapted from Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 23.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- The Rule of $\equiv -elimination$[11]

$$subst : (P : \mathbb{R} \to Set) \to \{x\ y : \mathbb{R}\} \to x \equiv y \to P\ x \to P\ y$$
$$subst\ P\ refl\ Px = Px$$

- The Rule of $\equiv -elimination$[11]

$$subst_2 : (P : \mathbb{R} \to \mathbb{R} \to Set) \to \{x_1\ x_2\ y_1\ y_2 : \mathbb{R}\} \to x_1 \equiv x_2 \to y_1 \equiv y_2 \to P\ x_1\ y_1 \to P\ x_2\ y_2$$
$$subst_2\ P\ refl\ refl\ h = h$$

[11] Adapted from Ana Bove and Peter Dybjer (2009). Dependent Types at Work, p. 23.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- $\forall a, b \in \mathbb{R}$, the relationship is introduced $" > "$ and $" < "$ Either the expression:

$$a > b \text{ or } b < a.$$

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- $\forall a, b \in \mathbb{R}$, the relationship is introduced $" > "$ and $" < "$ Either the expression:

$$a > b \text{ or } b < a.$$

- Where $a - b > 0$, also:

$$a \geq b \text{ or } b \leq a.$$

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- $\forall a, b \in \mathbb{R}$, the relationship is introduced $" > "$ and $" < "$ Either the expression:

$$a > b \text{ or } b < a.$$

- Where $a - b > 0$, also:

$$a \geq b \text{ or } b \leq a.$$

- That would be equivalent to: $a > b$ or $a = b$.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- Equational Reasoning[12]

$$\_\equiv\langle\_\rangle\_ : \forall\, x\ \{y\ z\} \to x \equiv y \to y \equiv z \to x \equiv z$$
$$\_\equiv\langle x \equiv y\rangle\ \ y \equiv z\ =\ \ \mathit{trans}\ \ x \equiv y\ \ y \equiv z$$

[12] Mu, S.-C., Ko, H.-S. and Jansson, P. (2009). Algebra of Programming in Agda: Dependent Types for Relational Program Derivation. Journal of Functional Programming 19.5, pp. 545-579.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Definitions

- Equational Reasoning[12]

$$\_\equiv\langle\_\rangle\_ : \forall\, x\ \{y\ z\} \to x \equiv y \to y \equiv z \to x \equiv z$$
$$\_\equiv\langle x \equiv y\rangle\ \ y \equiv z\ =\ \ \mathit{trans}\ \ x \equiv y\ \ y \equiv z$$

- $\_\blacksquare : \forall\, x \to x \equiv x$
  $\_\blacksquare\ \_ = \mathit{refl}$

[12] Mu, S.-C., Ko, H.-S. and Jansson, P. (2009). Algebra of Programming in Agda: Dependent Types for Relational Program Derivation. Journal of Functional Programming 19.5, pp. 545-579.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Conjuntion[13]

- $\textbf{data}\ \_\wedge\_\ (A\ B\ :\ \mathit{Set})\ :\ \mathit{Set}\ \mathit{where}$
  $\_,\_\ :\ A\ \rightarrow\ B\ \rightarrow\ A\ \wedge\ B$

[13] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, pp. 18-19.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Conjuntion[13]

- $\mathbf{data}\ \_\wedge\_\ (A\ B\ :\ Set)\ :\ Set\ where$
  $\_,\_\ :\ A\ \rightarrow\ B\ \rightarrow\ A\ \wedge\ B$

- $proj_1\ :\ \forall\ \{A\ B\}\ \rightarrow\ A\ \wedge\ B\ \rightarrow\ A$
  $proj_1\ (a\ ,\ \_)\ =\ a$

[13] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, pp. 18-19.

Real Numbers Formalization: Completeness Axiom and Automatic Reasoning

Conjuntion[13]

- $\mathbf{data}\ \_\wedge\_\ (A\ B\ :\ Set)\ :\ Set\ where$
  $\_,\_\ :\ A\ \rightarrow\ B\ \rightarrow\ A\ \wedge\ B$
- $proj_1\ :\ \forall\{A\ B\}\ \rightarrow\ A\ \wedge\ B\ \rightarrow\ A$
  $proj_1\ (a\ ,\ \_)\ =\ a$
- $proj_2\ :\ \forall\{A\ B\}\ \rightarrow\ A\ \wedge\ B\ \rightarrow\ B$
  $proj_2\ (\_\ ,\ b)\ =\ b$

[13] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, pp. 18-19.

Disjunction[14]

- $\mathbf{data}\ \_\vee\_\ (A\ B\ :\ Set)\ :\ Set\ where$
  $inj_1\ :\ A\ \rightarrow\ A\ \vee\ B$
  $inj_2\ :\ B\ \rightarrow\ A\ \vee\ B$

[14] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, pp. 19-20.

Disjunction[14]

- $\mathbf{data}\ \_\vee\_\ (A\ B\ :\ Set)\ :\ Set\ where$
  $inj_1\ :\ A\ \rightarrow\ A\ \vee\ B$
  $inj_2\ :\ B\ \rightarrow\ A\ \vee\ B$

- $case\ :\ \forall\ \{A\ B\}\ \rightarrow\ \{C\ :\ Set\}\ \rightarrow\ (A\ \rightarrow\ C)\ \rightarrow$
  $(B\ \rightarrow\ C)\ \rightarrow\ A\ \vee\ B\ \rightarrow\ C$
  $case\ f\ g\ (inj_1\ a)\ =\ f\ a$
  $case\ f\ g\ (inj_2\ b)\ =\ g\ b$

[14] Ana Bove and Peter Dybjer (2009). Dependent Types at Work, pp. 19-20.