Computer Formalization \ of the Real Numbers in Lean 4

HIIT Foundations Friday

Niklas Halonen

2025-09-26

Aalto University

ForAlli

The Aalto Formalization Alliance

Type Classes and API in Mathematics

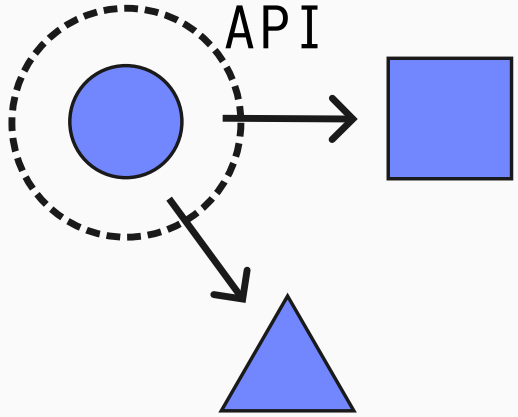
$$\mathbb{N} \overset{?}{\subset} \mathbb{Z} \overset{?}{\subset} \mathbb{Q} \overset{?}{\subset} \mathbb{R}$$

$$\mathbb{N} \overset{?}{\subset} \mathbb{Z} \overset{?}{\subset} \mathbb{Q} \overset{?}{\subset} \mathbb{R}$$

$$\text{Semigroup} \supseteq \text{Monoid} \supseteq \text{Group} \supseteq \text{Ring} \supseteq \text{Field}$$

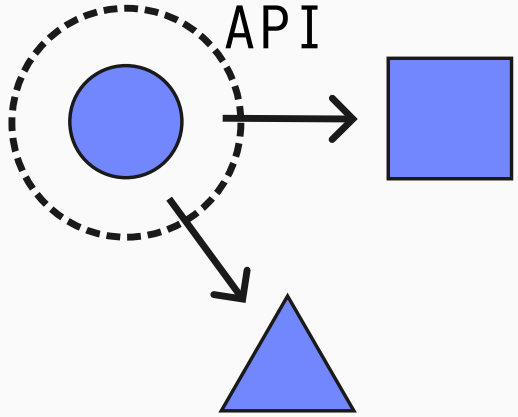
Type Classes and API in Mathematics

- API (Application Programming Interface) is the interface that is exposed by an object.
  - In **mathematics**: the object's *relationship* to other objects.
  - In **computer science**: the object's constructors, field accessors, methods.



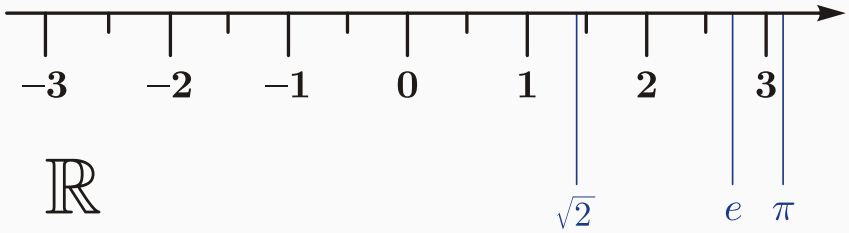
Type Classes and API in Mathematics

- API (Application Programming Interface) is the interface that is exposed by an object.
  - In **mathematics**: the object's *relationship* to other objects.
  - In **computer science**: the object's constructors, field accessors, methods.
- Shared behavior can be organized in **type classes**.
- In Mathlib (Lean's mathematics library) mathematical properties are expressed using a **type class hierarchy**.
  - Examples: `CommRing`, `Field`, `MetricSpace`.



What are Real numbers?

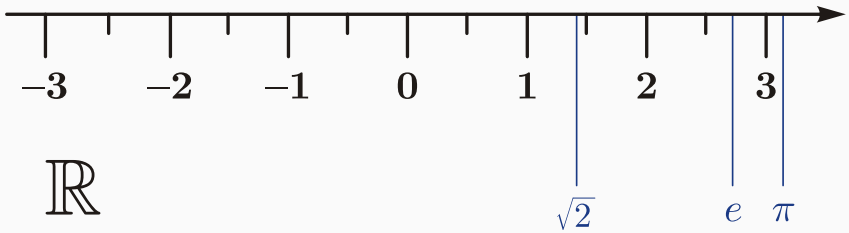
Informally



- The number line
- “Rationals ∪ irrationals”
- Convergent sequences have a limit

What are Real numbers?

Informally



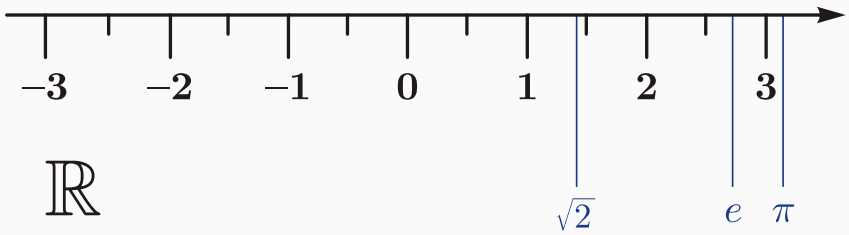
- The number line
- “Rationals ∪ irrationals”
- Convergent sequences have a limit

More formally

- R is a “conditionally complete linear ordered field”
  - *Conditionally complete*: every nonempty subset of R bounded from above has a least upper bound (in R).

What are Real numbers?

Informally



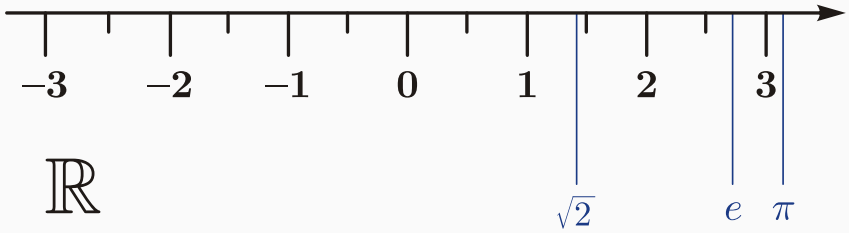
- The number line
- “Rationals ∪ irrationals”
- Convergent sequences have a limit

More formally

- R is a “conditionally complete linear ordered field”
  - *Conditionally complete*: every nonempty subset of R bounded from above has a least upper bound (in R).
  - *Linear ordered*: $x \leq y$ or $y \leq x$.

What are Real numbers?

Informally



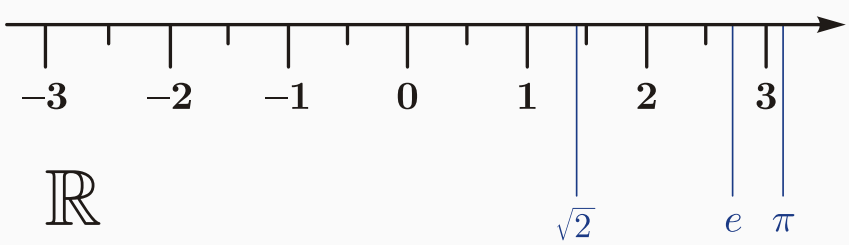
- The number line
- "Rationals ∪ irrationals"
- Convergent sequences have a limit

More formally

- R is a "conditionally complete linear ordered field"
  - *Conditionally complete*: every nonempty subset of R bounded from above has a least upper bound (in R).
  - *Linear ordered*: $x \leq y$ or $y \leq x$.
  - A *field*: R is a ring and each $x : R$ has a multiplicative inverse x^{-1}.

What are Real numbers?

Informally

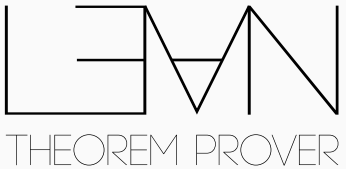


- The number line
- “Rationals ∪ irrationals”
- Convergent sequences have a limit

More formally

- R is a “conditionally complete linear ordered field”
  - *Conditionally complete*: every nonempty subset of R bounded from above has a least upper bound (in R).
  - *Linear ordered*: $x \leq y$ or $y \leq x$.
  - A *field*: R is a ring and each $x : R$ has a multiplicative inverse x^{-1}.
- Any such R is isomorphic with $\mathbb{R}$.

What is Computer Formalization?



Benefits of Computer Formalization

Organization

Verification

Automation

Benefits of Computer Formalization

Organization

Create **libraries** of intercompatible definitions and theorems.

Verification

Automation

Benefits of Computer Formalization

Organization

Create **libraries** of intercompatible definitions and theorems.

Verification

Proofs are systematically verified by a **trusted kernel**.

Automation

Benefits of Computer Formalization

Organization

Create **libraries** of intercompatible definitions and theorems.

Verification

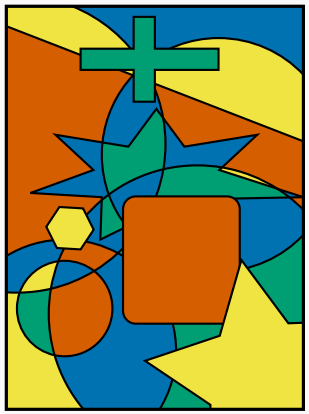
Proofs are systematically verified by a **trusted kernel**.

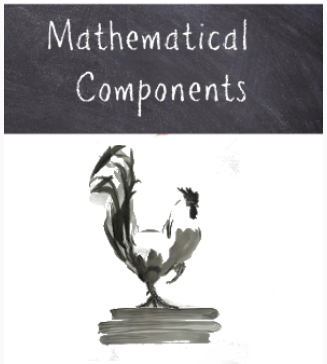
Automation

LLMs can verifiably prove theorems that **humans** care about.

2005 – 2012

Four-color and Feit-Thompson theorem

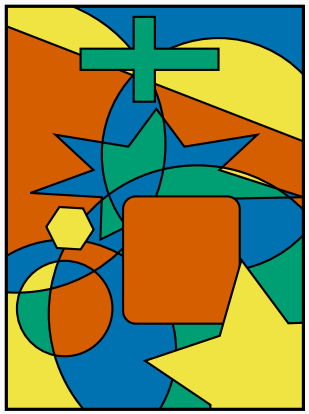


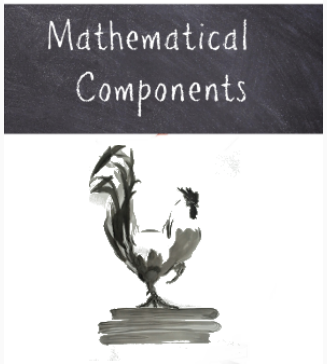


History of Computer Formalization

2005 – 2012

Four-color and Feit-Thompson theorem



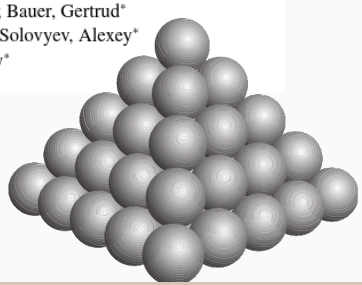


2014

Kepler conjecture

The Flyspeck project has been a large team effort over a period of years, and my contributions have been just a fraction of the whole. This appendix cites their contributions. The principal formalizer of each chapter is indicated with an asterisk.

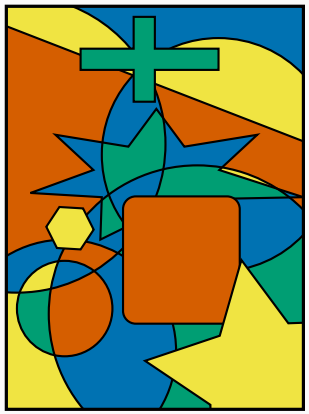
Text Chapter	**Author of formalization work**
Trigonometry	Nguyen Quang Truong*; Rute, Jason; Harrison, John; Vu Khac Ky
Volume	Harrison, John*; Nguyen Tat Thang
Hypermap	Tran Nam Trung*
Fan	Hoang Le Truong*; Harrison, John
Packing	Solovyev, Alexey*; Vu Khac Ky*; Nguyen Tat Thang; Hales, Thomas
Local Fan	Nguyen Quang Truong*; Hoang Le Truong*
Tame Hypermap	Solovyev, Alexey*; Dat Tat Dang; Trieu Thi Diep; Vu Quang Thanh; Vuong Anh Quyen
Code-Verification	**Author of formalization work**
Hypermap Generation	Nipkow, Tobias*; Bauer, Gertrud*
Linear Programs	Obua, Stephen*; Solovyev, Alexey*
Nonlinear Inequalities	Solovyev, Alexey*

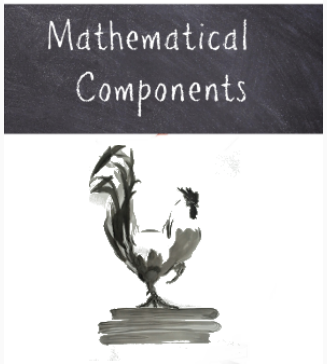


History of Computer Formalization

2005 – 2012

Four-color and Feit-Thompson theorem



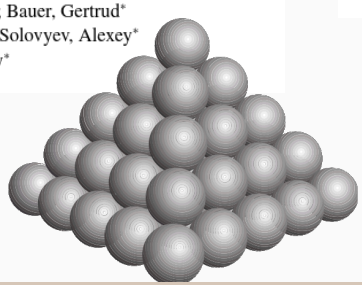


2014

Kepler conjecture

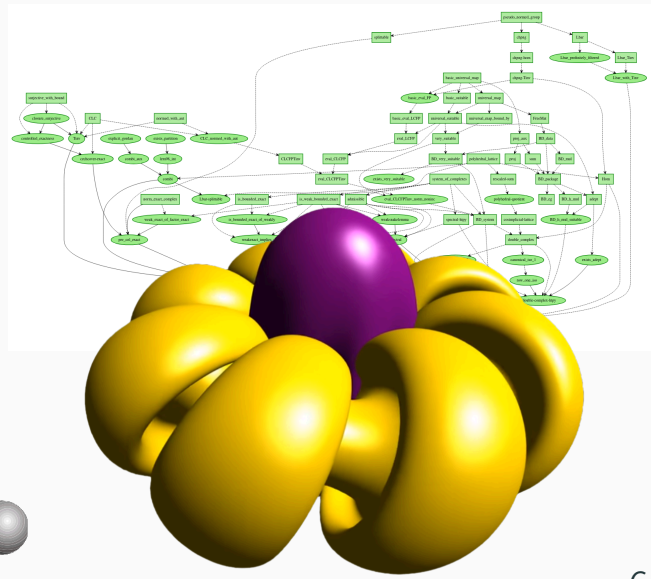
The Flyspeck project has been a large team effort over a period of years, and my contributions have been just a fraction of the whole. This appendix cites their contributions. The principal formalizer of each chapter is indicated with an asterisk.

Text Chapter	**Author of formalization work**
Trigonometry	Nguyen Quang Truong*; Rute, Jason; Harrison, John; Vu Khac Ky
Volume	Harrison, John*; Nguyen Tat Thang
Hypermap	Tran Nam Trung*
Fan	Hoang Le Truong*; Harrison, John
Packing	Solovyev, Alexey*; Vu Khac Ky*; Nguyen Tat Thang; Hales, Thomas
Local Fan	Nguyen Quang Truong*; Hoang Le Truong*
Tame Hypermap	Solovyev, Alexey*; Dat Tat Dang; Trieu Thi Diep; Vu Quang Thanh; Vuong Anh Quyen
Code-Verification	**Author of formalization work**
Hypermap Generation	Nipkow, Tobias*; Bauer, Gertrud*
Linear Programs	Obua, Stephen*; Solovyev, Alexey*
Nonlinear Inequalities	Solovyev, Alexey*



2019 –

Perfectoid Spaces, Liquid Tensor Experiment, Sphere Eversion, Mathlib



Formal Definition of the Reals

JSE *says:*

December 11, 2020 at 10:24 pm

Thanks for a very informative and idea-rich post, Peter!

Kevin, how are real numbers formalized in Lean in the first place? Via Cauchy sequences? If so, the idea of proving something about real vector spaces by exhibiting R as a quotient of a subring Z[[T]]_{>r} of Z[[T]] cut out by archimedean norm conditions does somehow seem in the spirit of things; indeed I suppose you could think of this description of R as an alternate means of formalizing the real numbers, one particularly well-suited for functional analysis!

★ Like

Reply

xenaproject *says:*

December 11, 2020 at 10:55 pm

Yes, real numbers are equivalence classes of Cauchy sequences under the hood, but of course the official API is that they're a complete ordered archimedean field and users are encouraged only to use that API when building new stuff.

the spirit of things; indeed I suppose you could think of this description of R as an alternate means of formalizing the real numbers, one particularly well-suited for functional analysis!

★ Like

Reply

xenaproject *says:* ← Kevin Buzzard

December 11, 2020 at 10:55 pm

Yes, real numbers are equivalence classes of Cauchy sequences under the hood, but of course the official API is that they're a complete ordered archimedean field and users are encouraged only to use that API when building new stuff.

[1]

the spirit of things; indeed I suppose you could think of this description of R as an alternate means of formalizing the real numbers, one particularly well-suited for functional analysis!

[1]

★ Like

Reply

xenaproject *says:* ← Kevin Buzzard

December 11, 2020 at 10:55 pm

Yes, real numbers are equivalence classes of Cauchy sequences under the hood, but of course the official API is that they're a complete ordered archimedean field and users are encouraged only to use that API when building new stuff.

the spirit of things; indeed I suppose you could think of this description of R as an alternate means of formalizing the real numbers, one particularly well-suited for functional analysis!

★ Like

Reply

xenaproject *says:* ← Kevin Buzzard

December 11, 2020 at 10:55 pm

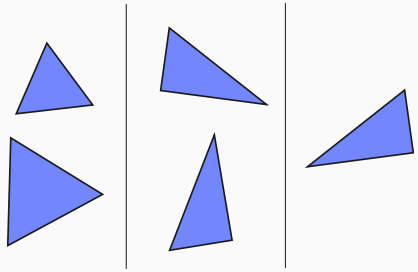
Yes, real numbers are equivalence classes of Cauchy sequences under the hood, but of course the official API is that they're a complete ordered archimedean field and users are encouraged only to use that API when building new stuff.

[1]

Quotients

Examples of quotients from mathematics:

- Congruent triangles
- Modular arithmetic, $\mathbb{Z}/3\mathbb{Z}$
- Real numbers, $\{f : \mathbb{N} \to \mathbb{Q} \mid f \text{ Cauchy}\}/\approx$
  - Where $f \approx g$ if they are "eventually ε-close"



Equivalence classes of congruent triangles

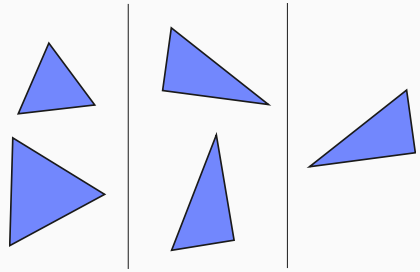
Quotients

Examples of quotients from mathematics:

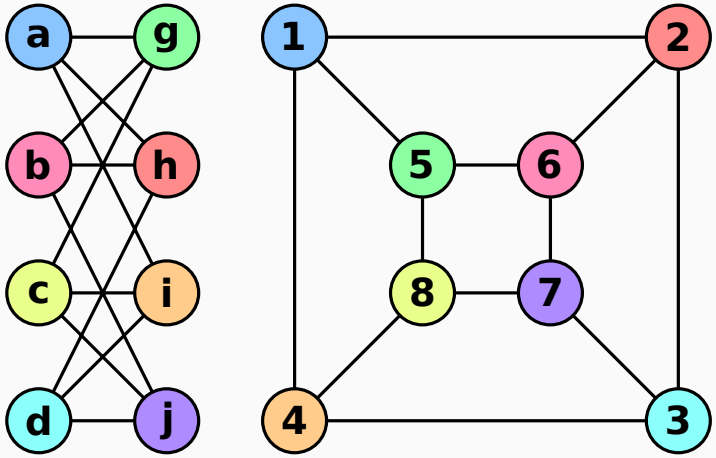
- Congruent triangles
- Modular arithmetic, $\mathbb{Z}/3\mathbb{Z}$
- Real numbers, $\{f : \mathbb{N} \to \mathbb{Q} \mid f \text{ Cauchy}\}/\approx$
  - Where $f \approx g$ if they are "eventually ε-close"

Examples of quotients from computer science:

- Multisets, a.k.a. lists modulo permutation
- Isomorphism classes of graphs: e.g. labeled graphs modulo relabeling of vertices



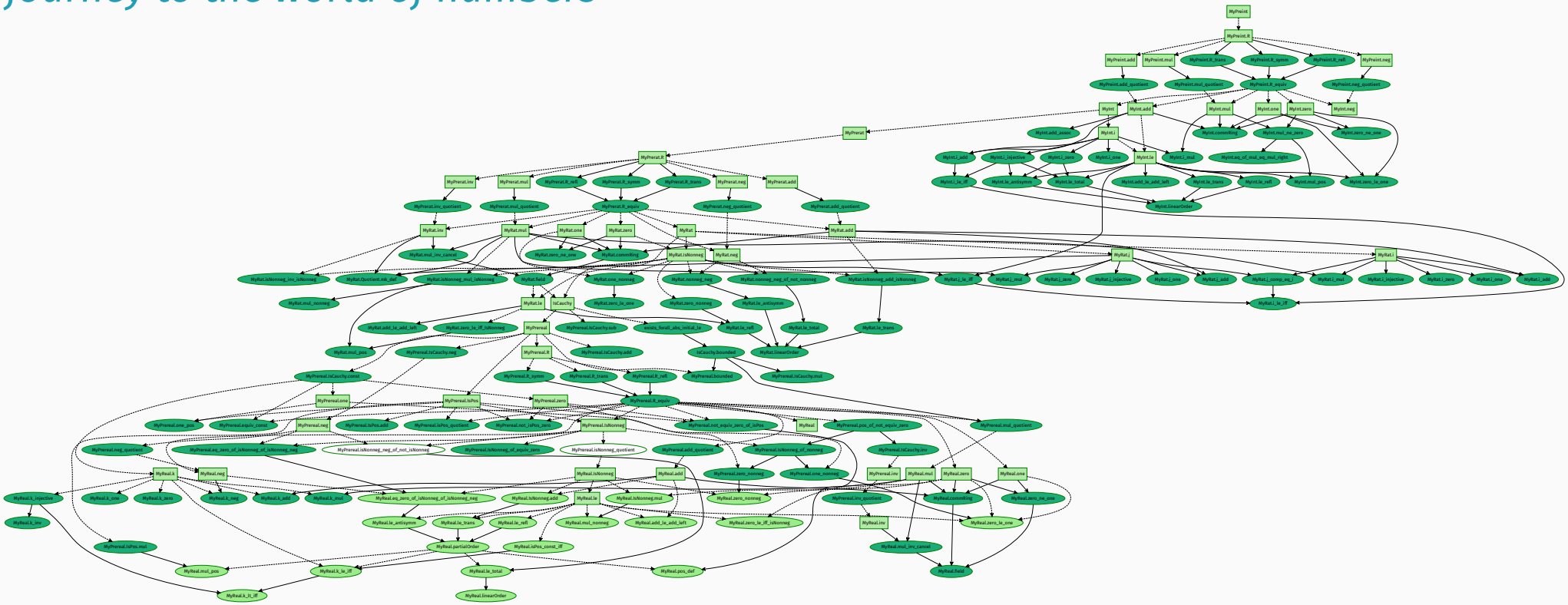
Equivalence classes of congruent triangles



Isomorphic graphs

Approach is built on top of Riccardo Brasca's

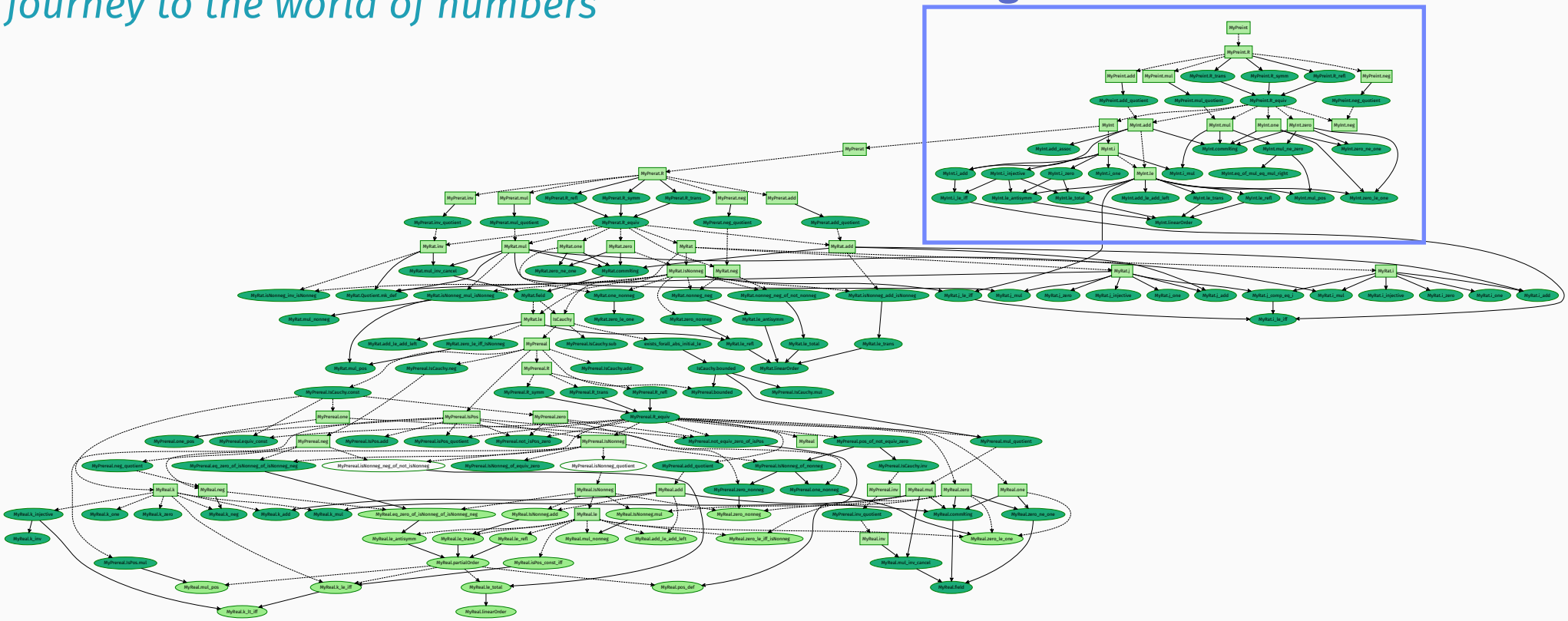
A journey to the world of numbers



Approach is built on top of Riccardo Brasca's

A journey to the world of numbers

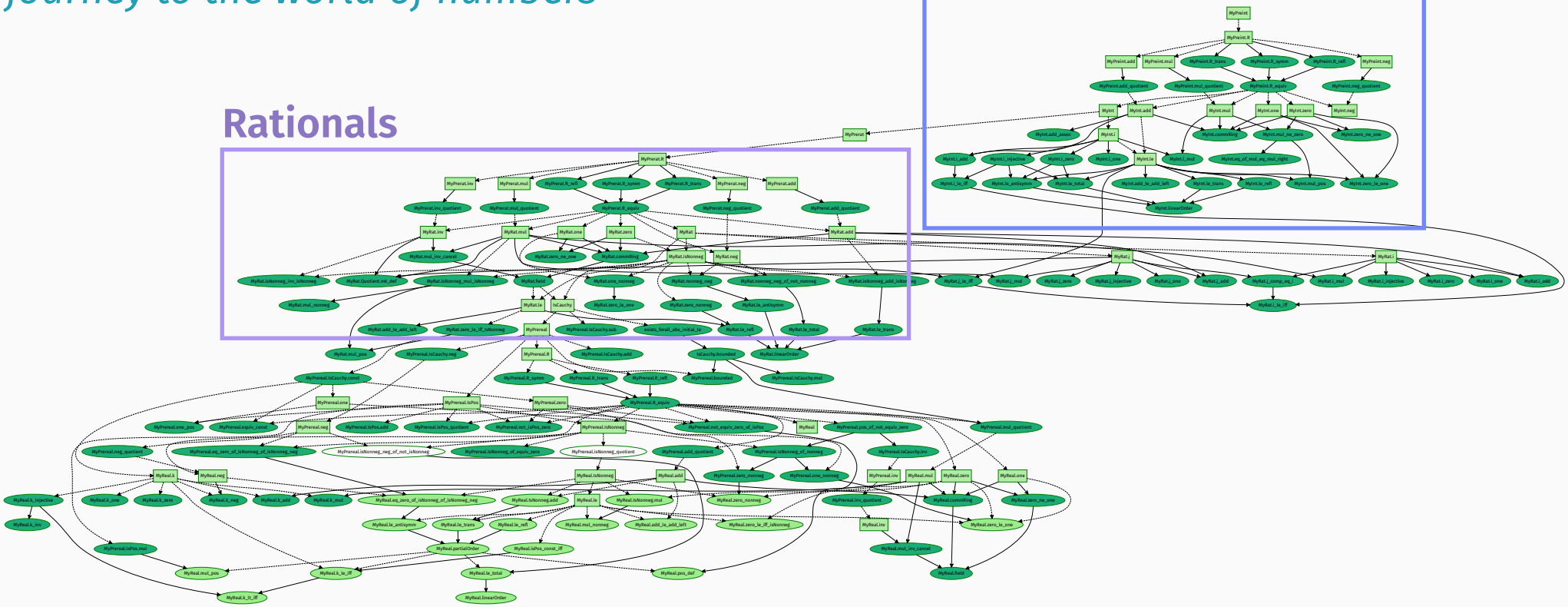
Integers



Approach is built on top of Riccardo Brasca's
A journey to the world of numbers

Integers

Rationals

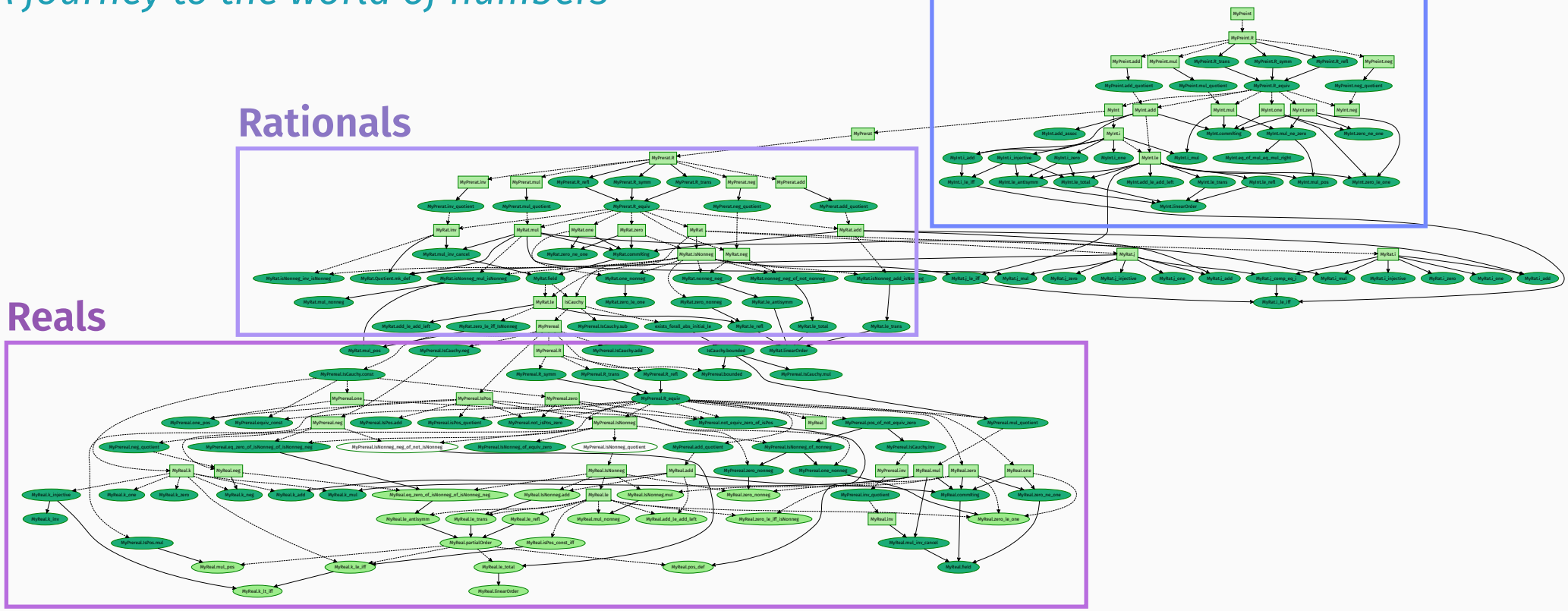


Approach is built on top of Riccardo Brasca's
A journey to the world of numbers

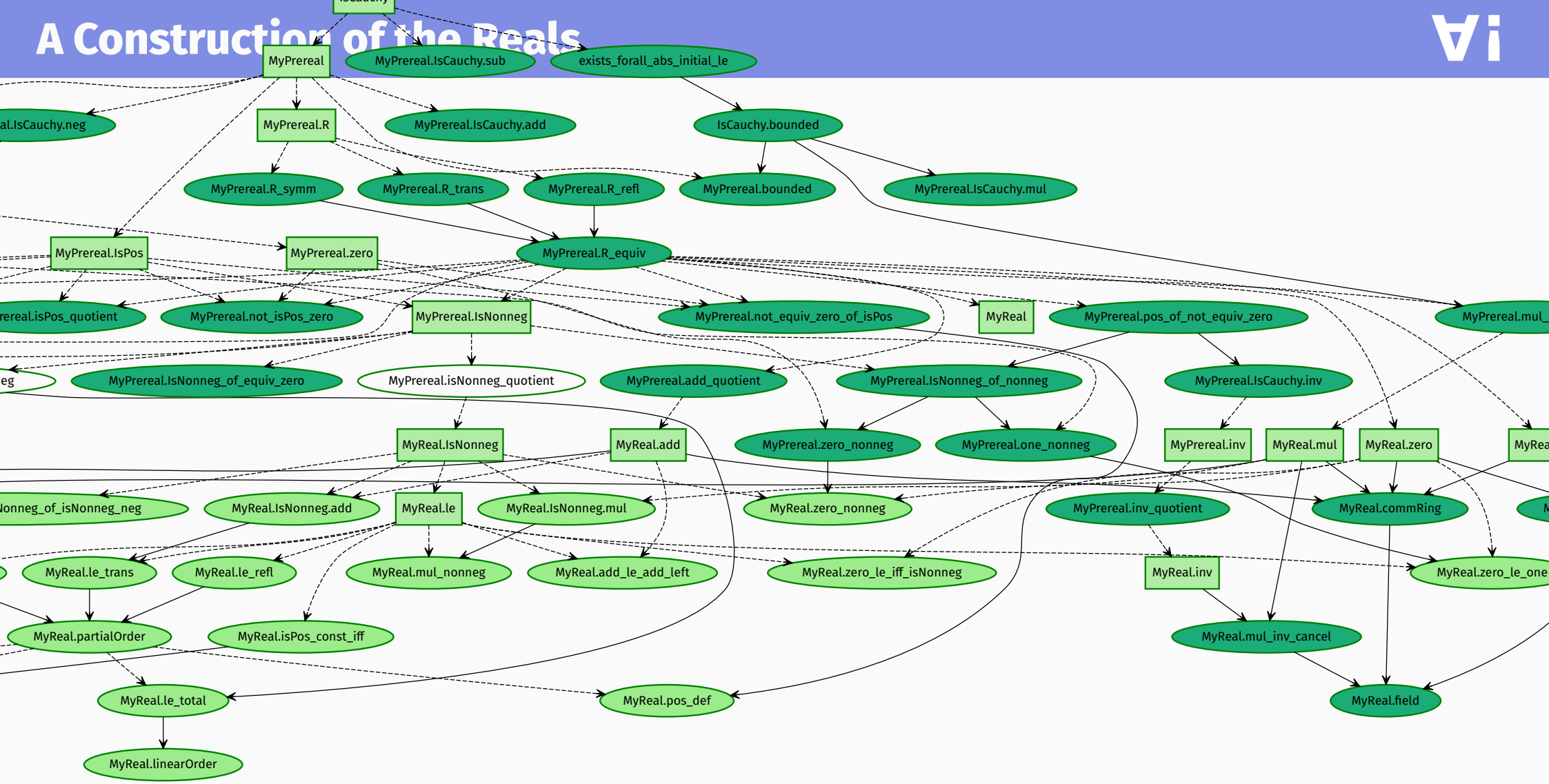
Integers

Rationals

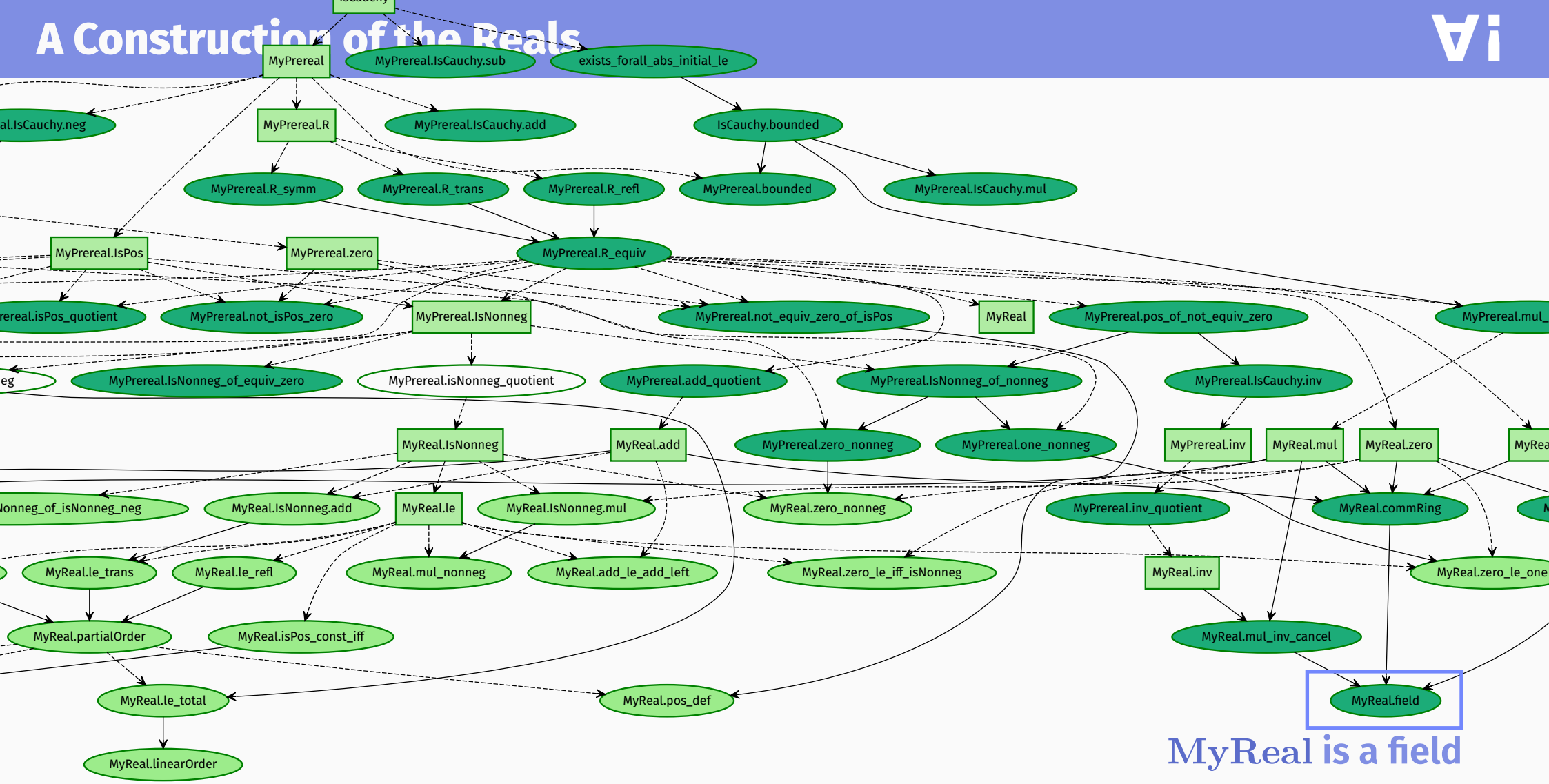
Reals



A Construction of the Reals

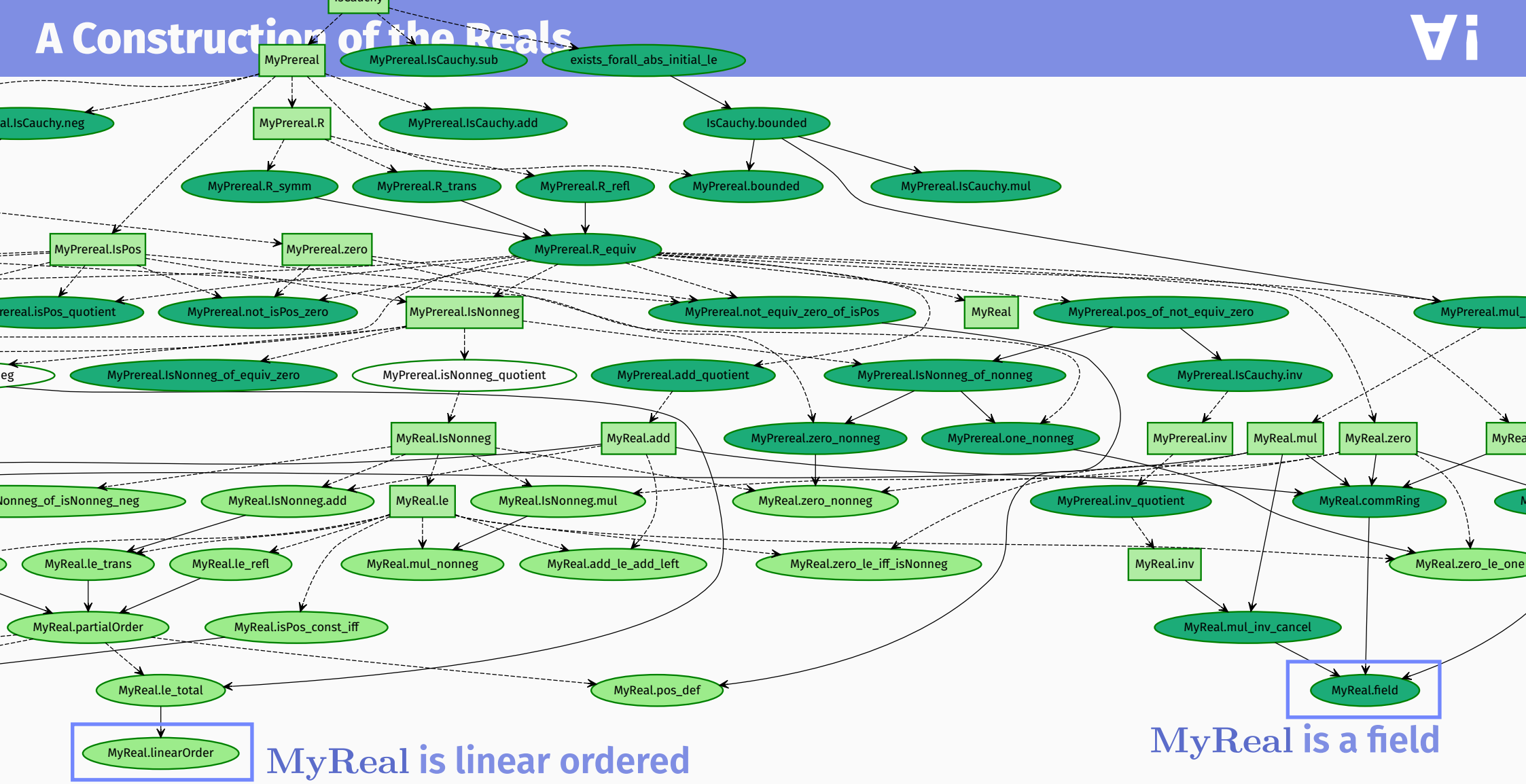


A Construction of the Reals



MyReal is a field

A Construction of the Reals



MyReal is linear ordered

MyReal is a field

- We can prove that our real numbers are **the real numbers**.
  - Mathlib gives the proof of isomorphism between all conditionally complete linear ordered fields:

```
LinearOrderedField.inducedOrderRingIso MyReal ℝ : MyReal ≃+*o ℝ
```

- Some API for real numbers, such as metric spaces (distances) is not immediately available for MyReal.
  - It is technically possible to use this for constructing the metric space structure, but I found it difficult to use in proofs of continuity for instance.

Metaprogramming and Tactics

Short Proof: Associativity of Addition

Let $x, y, z : \mathrm{MyReal}$, i.e. equivalence classes of Cauchy sequences. Addition is associative: $(x + y) + z = x + (y + z)$.

To show:

$$\underbrace{x = [\![a]\!]}_{a \text{ is a Cauchy sequnce}}$$

$$\vdash ([\![a]\!] + [\![b]\!]) + [\![c]\!] = [\![a]\!] + ([\![b]\!] + [\![c]\!]).$$

Proof: The equivalence classes are equal if their representatives are related.[1]

$$\vdash (a + b) + c \approx a + (b + c)$$

$$\vdash \forall \varepsilon > 0, \exists N, \forall n \geq N, |(a_n + b_n) + c_n - (a_n + (b_n + c_n))| \leq \varepsilon$$

Let $\varepsilon > 0$, use $N := 0$ and let $n \geq N$. Simplifying the left-hand side using ring axioms on rationals reduces the goal to $|0| \leq \varepsilon$, which follows from $\varepsilon > 0$.

[1] This is called the *quotient axiom* in Lean.

Short Proof: Associativity of Addition

Let $x, y, z : \mathrm{MyReal}$, i.e. equivalence classes of Cauchy sequences. Addition is associative: $(x + y) + z = x + (y + z)$.

To show:

$$\underbrace{x = [\![a]\!]}_{a \text{ is a Cauchy sequnce}}$$

$$\vdash ([\![a]\!] + [\![b]\!]) + [\![c]\!] = [\![a]\!] + ([\![b]\!] + [\![c]\!]).$$

Proof: The equivalence classes are equal if their representatives are related.[1]

$$\vdash (a + b) + c \approx a + (b + c)$$

$$\vdash \forall \varepsilon > 0, \exists N, \forall n \geq N, |(a_n + b_n) + c_n - (a_n + (b_n + c_n))| \leq \varepsilon$$

Let $\varepsilon > 0$, use $N := 0$ and let $n \geq N$. Simplifying the left-hand side using ring axioms on rationals reduces the goal to $|0| \leq \varepsilon$, which follows from $\varepsilon > 0$.

The other commutative ring axioms hold by a **"similar argument"**.

[1] This is called the *quotient axiom* in Lean.

```
macro "quot_proof" : tactic =>
  `(tactic|
  focus
    refine Quot.ind fun a => ?_
    try refine Quot.ind fun b => ?_
    try refine Quot.ind fun c => ?_
    apply Quot.sound
    intro ε hε
    simp
    use 0
    intro n hn
    try {ring_nf; simp [hε.le]}
  )
```

```
macro "quot_proof" : tactic =>
  `(tactic|
  focus
    refine Quot.ind fun a => ?_
    try refine Quot.ind fun b => ?_
    try refine Quot.ind fun c => ?_
    apply Quot.sound
    intro ε hε          ⟵ let ε > 0
    simp                ⟵ simplify
    use 0               ⟵ use N := 0
    intro n hn          ⟵ let n ≥ N
    try {ring_nf; simp [hε.le]}
  )
```

simplify with ring axioms

```
macro "quot_proof" : tactic =>
  `(tactic|
  focus
    refine Quot.ind fun a => ?_
    try refine Quot.ind fun b => ?_
    try refine Quot.ind fun c => ?_
    apply Quot.sound
    intro ε hε          ⟵ let ε > 0
    simp                ⟵ simplify
    use 0               ⟵ use N := 0
    intro n hn          ⟵ let n ≥ N
    try {ring_nf; simp [hε.le]}
  )
```

simplify with ring axioms

```
instance commRing : CommRing MyReal where
  add := (· + ·)
  add_assoc := by quot_proof
  zero := 0
  zero_add := by quot_proof
  add_zero := by quot_proof
  add_comm := by quot_proof
  mul := (· * ·)
  left_distrib := by quot_proof
  right_distrib := by quot_proof
  zero_mul := by quot_proof
  mul_zero := by quot_proof
  mul_assoc := by quot_proof
  one := 1
  one_mul := by quot_proof
  mul_one := by quot_proof
  neg := (- ·)
  mul_comm := by quot_proof
  neg_add_cancel := by quot_proof
```

Recap & Key points

1. API and type classes
   - It's useful to group shared behavior in type classes, even when formalizing mathematics. The theory can be generic (agnostic) over the construction.
     - However, concrete constructions are often easier to work with.

Recap & Key points

1. API and type classes
   - It's useful to group shared behavior in type classes, even when formalizing mathematics. The theory can be generic (agnostic) over the construction.
     - However, concrete constructions are often easier to work with.
2. Real numbers
   - We can what real numbers are in code, and get all of their properties.
   - Key "characteristic" properties can be expressed using type classes.
   - The existence of a construction proves that the theory of real numbers is consistent.

Recap & Key points

1. API and type classes
   - It's useful to group shared behavior in type classes, even when formalizing mathematics. The theory can be generic (agnostic) over the construction.
     - However, concrete constructions are often easier to work with.
2. Real numbers
   - We can what real numbers are in code, and get all of their properties.
   - Key "characteristic" properties can be expressed using type classes.
   - The existence of a construction proves that the theory of real numbers is consistent.
3. Metaprogramming
   - Proof that real numbers are a field boils down to field axioms on rationals.
     - We can make "cases are similar" arguments exact with computer formalization by defining a custom tactic.

Thank you for listening!

Questions?

Join the Aalto Formalization Alliance (ForAlli) for discussion.

Want to learn Lean? See these learning resources.

To contact me, send email to `niklas.2.halonen AT aalto.fi`.

Learning Resources From the Community Page

Learning Resource	Type	Audience	Level	Requirements
Natural Number Game	Exercises (game)	General	High-school	In-browser only
Lean game server (including NNG)	Exercises (game)	General	High-school	In-browser only
Mechanics of Proof	Book + Exercises	General	Undergraduate	Browser or local
GlimpseOfLean	Exercises	Math	Undergraduate math (1st year)	Browser is sufficient
Introduction to the Lean 4 theorem prover and programming language by Leonardo de Moura	Video	CS	Undergraduate CS (1st year)	YouTube
Lean Documentation Overview	Book	CS	Undergraduate CS	
Functional Programming in Lean (FPIL)	Book + Exercises	CS	Undergraduate CS (2nd year)	Local installation
Theorem Proving in Lean (TPIL)	Book + Exercises	CS	Undergraduate CS (3rd year)	Local installation
CS 99: Functional Programming and Theorem Proving in Lean 4	Course Slides + Exercises	CS + Math	Undergraduate and graduate	Local installation
Mathematics in Lean (MIL) mathematics_in_lean	Book + Exercises	Math	Undergraduate and graduate math	Local installation
Formalising Mathematics Handbook formalising-mathematics-notes	Notes + Exercises	Math	Undergraduate and graduate math	Local installation
The Hitchhiker's Guide to Logical Verification	Book + Exercises	CS	Undergraduate and graduate CS	Local installation

References

- Gonthier, Formal Proof - The Four-Color Theorem
- Feit, Thompson, Chapter I, from Solvability of groups of odd order, PACIFIC J. MATH, VOL. 13, NO. 3 (1963)
- Gonthier et al., A Machine-Checked Proof of the Odd Order Theorem
- Hales at al., A formal proof of the Kepler conjecture
- Hales, Dense Sphere Packings A Blueprint for Formal Proofs
- Paulson, Organizing Numerical Theories Using Axiomatic Type Classes
- Buzzard, Commelin, Massot, Lean perfectoid spaces
- Scholze, Liquid tensor experiment announcement
- Topaz, Definitions in the liquid tensor experiment
- Massot, Nash, van Doorn, Formalising the h-Principle and Sphere Eversion
- Lean 4 theorem prover
- The mathlib Community, The Lean Mathematical Library, mathlib4
- Spitters, van der Weegen, Type classes for mathematics in type theory. Mathematical Structures in Computer Science. 2011;21(4):795-825. doi:10.1017/S0960129511000119
- Baanen, Use and Abuse of Instance Parameters in the Lean Mathematical Library
- The Lean Language Reference, Quotients
- Brasca, A journey to the world of numbers
- Halonen, xhalo32/numbers