mathlib:
Lean's mathematical library

Johannes Hölzl

VU VRIJE UNIVERSITEIT AMSTERDAM

EUTYPES 2018, Aarhus

Introduction — what is mathlib

- (classical) mathematical library for Lean

Introduction — what is mathlib

- (classical) mathematical library for Lean
  classical – linear algebra, number theory, analysis, ...

Introduction — what is mathlib

- (classical) mathematical library for Lean
  classical – linear algebra, number theory, analysis, ...
  classical – using choice and LEM

Introduction — what is mathlib

- (classical) mathematical library for Lean
  classical – linear algebra, number theory, analysis, ...
  classical – using choice and LEM
- Formerly distributed with Lean itself
  Leo wanted more flexibility

Introduction — what is mathlib

- (classical) mathematical library for Lean
  classical – linear algebra, number theory, analysis, ...
  classical – using choice and LEM
- Formerly distributed with Lean itself
  Leo wanted more flexibility
- Some (current) topics:
  Analysis, Linear Algebra, Set Theory, Number Theory, ...

Introduction — what is `mathlib`

- (classical) mathematical library for Lean
  classical – linear algebra, number theory, analysis, ...
  classical – using choice and LEM
- Formerly distributed with Lean itself
  Leo wanted more flexibility
- Some (current) topics:
  Analysis, Linear Algebra, Set Theory, Number Theory, ...
- Goal: be comparable to
  Coq's mathematical components,
  Isabelle's `HOL-Analysis`

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)
- Currently maintained by Mario Carneiro and me

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)
- Currently maintained by Mario Carneiro and me
- Some Contributors:

  Chris Hughes, Gabriel Ebner, Jeremy Avigad, Johann Commelin, Johannes Hölzl, Keely Hoek, Kenny Lau, Kevin Buzzard, Mario Carneiro, Minchao Wu, Patrick Massot, Reid Barton, Robert Y. Lewis, Scott Morrison, Sean Leather, Sebastian Ullrich, Sebastien Gouezel, Simon Hudon, ...

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)
- Currently maintained by Mario Carneiro and me
- Some Contributors:

  Chris Hughes, Gabriel Ebner, Jeremy Avigad, Johann Commelin, Johannes Hölzl, Keely Hoek, Kenny Lau, Kevin Buzzard, Mario Carneiro, Minchao Wu, Patrick Massot, Reid Barton, Robert Y. Lewis, Scott Morrison, Sean Leather, Sebastian Ullrich, Sebastien Gouezel, Simon Hudon, ...
- Projects:

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)
- Currently maintained by Mario Carneiro and me
- Some Contributors:

  Chris Hughes, Gabriel Ebner, Jeremy Avigad, Johann Commelin, Johannes Hölzl, Keely Hoek, Kenny Lau, Kevin Buzzard, Mario Carneiro, Minchao Wu, Patrick Massot, Reid Barton, Robert Y. Lewis, Scott Morrison, Sean Leather, Sebastian Ullrich, Sebastien Gouezel, Simon Hudon, ...
- Projects:

  Xena Project by Kevin Buzzard – teaching Lean to math students

Maintenance

- Currently $\sim$ 92.000 LOC (from $\sim$ 54.000 in Jan 2018)
- Currently maintained by Mario Carneiro and me
- Some Contributors:

  Chris Hughes, Gabriel Ebner, Jeremy Avigad, Johann Commelin, Johannes Hölzl, Keely Hoek, Kenny Lau, Kevin Buzzard, Mario Carneiro, Minchao Wu, Patrick Massot, Reid Barton, Robert Y. Lewis, Scott Morrison, Sean Leather, Sebastian Ullrich, Sebastien Gouezel, Simon Hudon, ...

- Projects:

  Xena Project by Kevin Buzzard – teaching Lean to math students
  Lean Forward by Jasmin C. Blanchette – number theory

Logic

- Lean is (mostly) CIC

Logic

- Lean is (mostly) CIC
- Proof irrelevance is a definitional equality (incompatible with HoTT)

Logic

- Lean is (mostly) CIC
- Proof irrelevance is a definitional equality (incompatible with HoTT)
- Quotient types (built-in)

Logic

- Lean is (mostly) CIC
- Proof irrelevance is a definitional equality (incompatible with HoTT)
- Quotient types (built-in)
- Non-cummulative universes

Logic

- Lean is (mostly) CIC
- Proof irrelevance is a definitional equality (incompatible with HoTT)
- Quotient types (built-in)
- Non-cummulative universes
- `axiom choice` : $\Pi(\alpha : \texttt{Sort}_u), \texttt{nonempty}\ \alpha \to \alpha$

Logic

- Lean is (mostly) CIC
- Proof irrelevance is a definitional equality (incompatible with HoTT)
- Quotient types (built-in)
- Non-cummulative universes
- `axiom choice` : $\Pi(\alpha : \texttt{Sort}_u), \texttt{nonempty}\ \alpha \to \alpha$
- `noncomputable`: constant uses axioms, after `Prop` erasure (many things are computable)

mathlib

- Constructions
- Tactics
- Theories

Structures as Complete Lattices

There are many `set` like structures:

`filter`, `topology`, `uniformity`, `submodule`, `measurable`

Structures as Complete Lattices

There are many `set` like structures:

`filter`, `topology`, `uniformity`, `submodule`, `measurable`

- Galois insertion relates them to another complete lattice, e.g. `set` α:

$$\begin{aligned} \texttt{generate} &: \texttt{set}\ \alpha \to \mathcal{F}\ \alpha \\ \texttt{forget} &: \mathcal{F}\ \alpha \to \texttt{set}\ \alpha \end{aligned}$$

$$\texttt{generate}\ s \le x \longleftrightarrow s \subseteq \texttt{forget}\ x$$
$$\texttt{generate} \circ \texttt{forget} = id$$

Structures as Complete Lattices

There are many `set` like structures:

`filter`, `topology`, `uniformity`, `submodule`, `measurable`

- Galois insertion relates them to another complete lattice, e.g. `set` α:

$$\begin{aligned} \texttt{generate} &: \texttt{set}\ \alpha \to \mathcal{F}\ \alpha \\ \texttt{forget} &: \mathcal{F}\ \alpha \to \texttt{set}\ \alpha \end{aligned}$$

$$\begin{aligned} &\texttt{generate}\ s \le x \longleftrightarrow s \subseteq \texttt{forget}\ x \\ &\texttt{generate} \circ \texttt{forget} = id \end{aligned}$$

- Complete lattices structure is lifted along these insertion:

$$\texttt{complete\_lattice}\ (\mathcal{F}\ \alpha)$$

Structures as Complete Lattices

There are many `set` like structures:

$$\texttt{filter}, \texttt{topology}, \texttt{uniformity}, \texttt{submodule}, \texttt{measurable}$$

- Galois insertion relates them to another complete lattice, e.g. $\texttt{set}\ \alpha$:

$$\begin{aligned} \texttt{generate} &: \texttt{set}\ \alpha \to \mathcal{F}\ \alpha \\ \texttt{forget} &: \mathcal{F}\ \alpha \to \texttt{set}\ \alpha \end{aligned}$$

$$\begin{aligned} &\texttt{generate}\ s \le x \longleftrightarrow s \subseteq \texttt{forget}\ x \\ &\texttt{generate} \circ \texttt{forget} = id \end{aligned}$$

- Complete lattices structure is lifted along these insertion:

$$\texttt{complete\_lattice}\ (\mathcal{F}\ \alpha)$$

- No category theory yet!

Complete Lattices with (co)functors

- These structures are co- and contravariant functors:

$$\texttt{map} : (\alpha \to \beta) \to (\mathcal{F}\ \alpha \to \mathcal{F}\ \beta)$$
$$\texttt{comap} : (\alpha \to \beta) \to (\mathcal{F}\ \beta \to \mathcal{F}\ \alpha)$$

  Also known as push-forward and pull-back.

Complete Lattices with (co)functors

- These structures are co- and contravariant functors:

$$\texttt{map} : (\alpha \to \beta) \to (\mathcal{F}\ \alpha \to \mathcal{F}\ \beta)$$
$$\texttt{comap} : (\alpha \to \beta) \to (\mathcal{F}\ \beta \to \mathcal{F}\ \alpha)$$

  Also known as push-forward and pull-back.
- They form a Galois connection:

$$\texttt{map}\ f\ x \leq y \longleftrightarrow x \leq \texttt{comap}\ f\ y$$

$$f \in \mathit{Hom}(x, y) := x \leq \texttt{comap}\ f\ y$$

Complete Lattices with (co)functors

- These structures are co- and contravariant functors:

$$\texttt{map} : (\alpha \to \beta) \to (\mathcal{F}\ \alpha \to \mathcal{F}\ \beta)$$
$$\texttt{comap} : (\alpha \to \beta) \to (\mathcal{F}\ \beta \to \mathcal{F}\ \alpha)$$

  Also known as push-forward and pull-back.
- They form a Galois connection:

$$\texttt{map}\ f\ x \leq y \longleftrightarrow x \leq \texttt{comap}\ f\ y$$

$$f \in \mathit{Hom}(x, y) := x \leq \texttt{comap}\ f\ y$$

- Easy constructions:

$$\texttt{prod}\ t_1\ t_2 := \texttt{comap}\ \pi_1\ t_1 \sqcup \texttt{comap}\ \pi_2\ t_2$$
$$\texttt{subtype}\ t\ s := \texttt{comap}\ (\texttt{subtype.val}\ s)\ t$$

Constructions – Adjoin $\top$, $\bot$ or 0

- Ext. nonnneg. reals: $\mathbb{R}_{\geq 0} \uplus \infty$, extended nats $\overline{\mathbb{N}} = \mathbb{N} \uplus \infty$
- Multisets with infinity (e.g. sets of factors, ∞ to represent 0)
- `with_top` $\alpha :=$ `option` α
- Instances:
  - `partial_order` $\alpha \to$ `partial_order` (`with_top` α)
  - `conditionally_complete_lattice_bot` $\alpha \to$ `complete_lattice` (`with_top` α)
  - `add_monoid` $\alpha \to$ `add_monoid` (`with_top` α)
  - `canonically_ordered_comm_semiring` $\alpha \to$ `canonically_ordered_comm_semiring` (`with_top` α)

Tactics

ring, abel (Mario Carneiro)
Decides ring / group equalities.

linarith (Robert Y. Lewis)
Decides linear problems (based Fourier-Motzkin elimination)

tidy (Scott Morrison)
Apply a list of tactics (suggests a replacement)

mathlib

- Basic (computable) data
- Type class hierarchies:
  - **Orders** orders, lattices
  - **Algebraic** (commutative) groups, rings, fields
  - **Spaces** measurable, topological, uniform, metric
- Cardinals & ordinals
- Analysis: topology, measure Theory, …
- Linear algebra
- Group / ring / field theory
- Number theory

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, $\mathsf{fin} : \mathbb{N} \to \mathsf{Type}$

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, fin $: \mathbb{N} \to$ `Type`
- `set` $\alpha := \alpha \to$ `Prop`

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, fin $: \mathbb{N} \to$ `Type`
- `set` $\alpha := \alpha \to$ `Prop`
- `list` α

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, fin : $\mathbb{N} \to$ Type
- set $\alpha := \alpha \to$ Prop
- list α
- multiset $\alpha :=$ list $\alpha/_{\text{perm}}$

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, fin : $\mathbb{N} \to$ Type
- set $\alpha := \alpha \to$ Prop
- list α
- multiset $\alpha :=$ list $\alpha/_{\text{perm}}$
- finset $\alpha := \{m : \text{multiset } \alpha \mid \text{nodup } m\}$

Basic (computable) data

- Numbers: $\mathbb{N}$, $\mathbb{Z}$ (as datatype, not quotient), $\mathbb{Q}$, fin : $\mathbb{N} \to$ Type
- set $\alpha := \alpha \to$ Prop
- list α
- multiset $\alpha :=$ list $\alpha/_{\text{perm}}$
- finset $\alpha := \{m : \text{multiset } \alpha \mid \text{nodup } m\}$
- Big operators for list, multiset and finset

Definition (Equivalence, Cardinals and Ordinals)

$$\texttt{structure}\ \alpha \simeq \beta := (f : \alpha \to \beta)\ (g : \beta \to \alpha)\ (f\_g : f \circ g = id)\ (g\_f : g \circ f = id)$$
$$\texttt{cardinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Type}_u/\simeq$$
$$\texttt{ordinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Well\_order}_u/\simeq_{ord}$$

- Well-ordered, semiring, etc...

Definition (Equivalence, Cardinals and Ordinals)

$$\texttt{structure}\ \alpha \simeq \beta := (f : \alpha \to \beta)\ (g : \beta \to \alpha)\ (f\_g : f \circ g = id)\ (g\_f : g \circ f = id)$$

$$\texttt{cardinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Type}_u/\simeq$$

$$\texttt{ordinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Well\_order}_u/\simeq_{ord}$$

- Well-ordered, semiring, etc...
- Semiring structure of `cardinal` from $\simeq$ constructions

Definition (Equivalence, Cardinals and Ordinals)

$$\texttt{structure}\ \alpha \simeq \beta := (f : \alpha \to \beta)\ (g : \beta \to \alpha)\ (f\_g : f \circ g = id)\ (g\_f : g \circ f = id)$$

$$\texttt{cardinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Type}_u/\simeq$$

$$\texttt{ordinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Well\_order}_u/\simeq_{ord}$$

- Well-ordered, semiring, etc...
- Semiring structure of `cardinal` from $\simeq$ constructions
- $\kappa + \kappa = \kappa = \kappa * \kappa$ (for $\kappa \geq \omega$)

Definition (Equivalence, Cardinals and Ordinals)

```
structure α ≃ β := (f : α → β) (g : β → α) (f_g : f ∘ g = id) (g_f : g ∘ f = id)
cardinal_u  : Type_{u+1} := Type_u/≃
ordinal_u   : Type_{u+1} := Well_order_u/≃_ord
```

- Well-ordered, semiring, etc...
- Semiring structure of `cardinal` from $\simeq$ constructions
- $\kappa + \kappa = \kappa = \kappa * \kappa$ (for $\kappa \geq \omega$)
- Existence of inaccessible cardinals (i.e. in the next universe)

Definition (Equivalence, Cardinals and Ordinals)

$$\texttt{structure}\ \alpha \simeq \beta := (f : \alpha \to \beta)\ (g : \beta \to \alpha)\ (f\_g : f \circ g = id)\ (g\_f : g \circ f = id)$$
$$\texttt{cardinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Type}_u/\simeq$$
$$\texttt{ordinal}_u \ : \ \texttt{Type}_{u+1} \ := \ \texttt{Well\_order}_u/\simeq_{ord}$$

- Well-ordered, semiring, etc...
- Semiring structure of `cardinal` from $\simeq$ constructions
- $\kappa + \kappa = \kappa = \kappa * \kappa$ (for $\kappa \geq \omega$)
- Existence of inaccessible cardinals (i.e. in the next universe)
- **Application:** Dimension of subspaces

p-adic numbers $\mathbb{Q}_p$ by Rob Y. Lewis

- Cauchy construction of $\mathbb{Q}$ "in the other direction"

$$n.d_0d_1d_2d_3\cdots \quad \mathbb{R}$$
$$\cdots d_4d_3d_2d_1d_0 \quad p\text{-adic number, } d_n < p$$

- Interesting result: Hensel's lemma

Quadratic Reciprocity by Chris Hughes ($\sim$ 1300 lines patch)

```
theorem quadratic_reciprocity (hp : prime p) (hq : prime q)
  (hp1 : p % 2 = 1) (hq1 : q % 2 = 1) (hpq : p ≠ q) :
  legendre p q hq * legendre q p hp = (-1) ^ (p/2 * q/2)
```

Filter generalizes limits (derived from Isabelle/HOL)

Analysis

Filter generalizes limits (derived from Isabelle/HOL)

Topology nhds filter, open & closed & compact sets, interior, closure

Analysis

Filter generalizes limits (derived from Isabelle/HOL)

Topology nhds filter, open & closed & compact sets, interior, closure

Uniformity complete, totally bounded, completion

`compact` $\leftrightarrow$ `complete` & `totally bounded`

Analysis

Filter generalizes limits (derived from Isabelle/HOL)

Topology nhds filter, open & closed & compact sets, interior, closure

Uniformity complete, totally bounded, completion
`compact` $\leftrightarrow$ `complete` & `totally bounded`

Metric instance of uniformities

Analysis

Filter generalizes limits (derived from Isabelle/HOL)

Topology nhds filter, open & closed & compact sets, interior, closure

Uniformity complete, totally bounded, completion
`compact` $\leftrightarrow$ `complete` & `totally bounded`

Metric instance of uniformities

Norm only rudimentary (no deriviatives yet...)

Analysis

Filter generalizes limits (derived from Isabelle/HOL)

Topology nhds filter, open & closed & compact sets, interior, closure

Uniformity complete, totally bounded, completion
`compact` $\leftrightarrow$ `complete` & `totally bounded`

Metric instance of uniformities

Norm only rudimentary (no deriviatives yet...)

Measure Lebesgue measure etc...

(Outer) Measures

```
structure outer_measure (α : Type*) :=
(μ : set α → ennreal) ...

structure measure (α) [measurable_space α]
  extends outer_measure α := ...
```

- Outer measures provide natural totalization of measures
- Carathéodory's extension theorem
- Lebesgue Measure
- Completion measurable space

```
def pmf (α) := { f : α → nnreal // is_sum f 1 }

def pure (a : α) : pmf α :=
⟨λa', if a' = a then 1 else 0, is_sum_ite _ _⟩

def bind (p : pmf α) (f : α → pmf β) : pmf β :=
⟨λb, (∑a, p a * f a b), ...⟩
```

- prove rules of the Giry monad
- generality of $\sum$ helps!

Definition (Infinite sum)

$$\sum_{i:\iota} f\ i \quad = \quad \lim_{s \to \texttt{at\_top}} \sum_{i \in s} f\ i$$

Assuming: $f : \iota \to \alpha$, [topological_add_monoid α]

- at_top : filter (finset ι) finite sets approaching univ : set ι
- $\mathbb{R}$, normed vector spaces, ennreal, $\mathbb{Q}$, $\mathbb{N}$, $\mathbb{Z}$, ...
- $\sum_b \sum_c f\ (b, c) = \sum_{(b,c)} f\ (b, c)$ — α regular
- $\sum_{n:\mathbb{N}} f\ n = \lim_{i \to \infty} \sum_{n=0}^{i} f\ n$

Definition (Module)

```
class module (α : out Type u) (β : Type v) [out ring α] := ...
```

Constructions: Subspace, Linear maps, Quotient, Product, Tensor Product

Dimension: `dim (α β) [field α] [vector_space α β] : cardinal`

Laws:

$$\frac{dom(f)}{ker(f)} \simeq_\ell im(f) \qquad \frac{s}{s \cap t} \simeq_\ell \frac{s \oplus t}{t}$$

$$R \oplus M \simeq_\ell M \qquad M \oplus N \simeq_\ell N \oplus M \qquad M \oplus (N \oplus L) \simeq_\ell (M \oplus N) \oplus L$$

Discussion

```
class module (α : out Type u) (β : Type v) [out ring α] :=
...
```

```
class module (α : out Type u) (β : Type v) [out ring α] :=
...
```

- Type class mechanism looks for `module _ β _`

```
class module (α : out Type u) (β : Type v) [out ring α] :=
...
```

- Type class mechanism looks for `module _ β _`
- Only one canoncial `module` per type

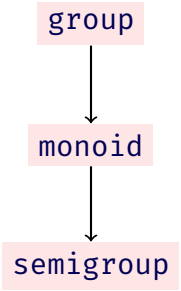
```
class module (α : out Type u) (β : Type v) [out ring α] :=
...
```

- Type class mechanism looks for `module _ β _`
- Only one canoncial `module` per type
- Idea: usually α is fixed per theory anyway

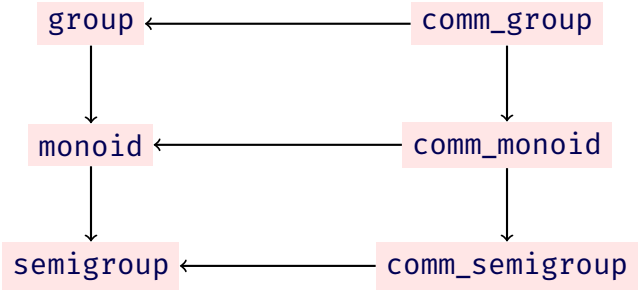
```
class module (α : out Type u) (β : Type v) [out ring α] :=
...
```

- Type class mechanism looks for `module _ β _`
- Only one canoncial `module` per type
- Idea: usually α is fixed per theory anyway
- **Problem:** (multivariate) polynomials, $\mathbb{Z}$-modules, $\mathbb{N}$-semimodules, ...

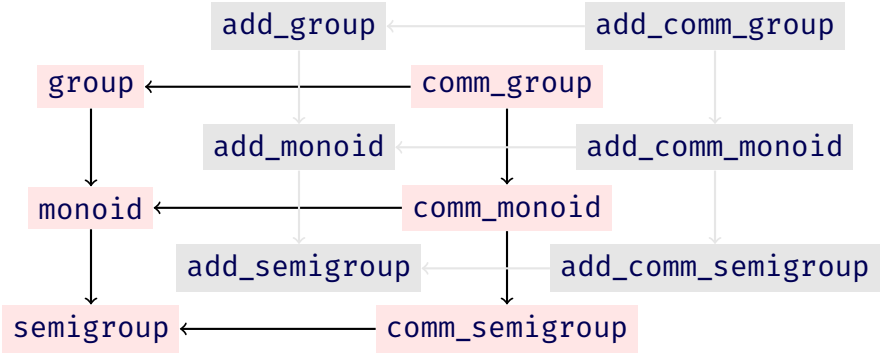
Problems with Type Classes



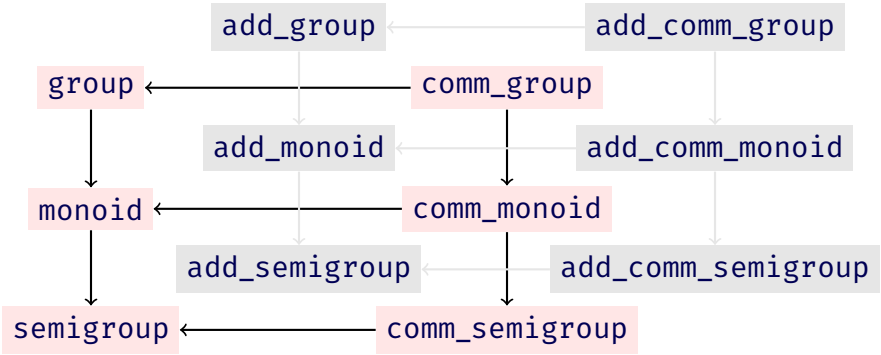
Problems with Type Classes



Problems with Type Classes



Problems with Type Classes



- Currently a automated copy from `group` to `add_group`
  instead: [*is_group*$(*)(/)(\square^{-1})1$] and [*is_group*$(+)(-)(-\square)0$]
- Mixin type classes
  replace `comm_monoid`, … by [*is_commutative* $(*)$]

Problem with Universes

$$
\begin{aligned}
&\texttt{class functor } (M : \texttt{Type}_u \to \texttt{Type}_v) := \\
&(\texttt{map} : \forall (\alpha\ \beta : \texttt{Type}_u), (\alpha \to \beta) \to M\ \alpha \to M\ \beta) \\
&(\texttt{map\_comp} : \forall (\alpha\ \beta\ \gamma : \texttt{Type}_u)\ f\ g\ h, \texttt{map}\ f \circ \texttt{map}\ g = \texttt{map}\ (f \circ g)) \\
&(\texttt{map\_id} : \forall \alpha, \texttt{map id} = \texttt{id})
\end{aligned}
$$

Problem with Universes

Problematic u

```
class functor (M : Type_u → Type_v) :=
(map : ∀(α β : Type_u), (α → β) → M α → M β)
(map_comp : ∀(α β γ : Type_u) f g h, map f ∘ map g = map (f ∘ g))
(map_id : ∀α, map id = id)
```

Problem with Universes

Problematic u

```
class functor (M : Type_u → Type_v) :=
(map : ∀(α β : Type_u), (α → β) → M α → M β)
(map_comp : ∀(α β γ : Type_u) f g h, map f ∘ map g = map (f ∘ g))
(map_id : ∀α, map id = id)
```

If we only work with `functor (topology α)` our library is too limited, e.g. `topology.map` allows mapping between different universes.

```
topology.map {α : Type u} {β : Type v} :
  (α → β) → (topology α → topology β)
```

Future

- PR: Trigonometric functions
- PR: Sylow's theorem
- Integral & Derivatives
- Category Theory
- Lean 4

mathlib

A (classical) mathematical library for Lean

https://github.com/leanprover/mathlib