Competing Inheritance Paths in Dependent Type Theory

a case study in functional analysis

Reynald Affeldt, Cyril Cohen, Marie Kerjean,
Assia Mahboubi, Damien Rouhling, Kazuhiko Sakaguchi

AIST / Inria / U. Tsukuba

2020-07-01

What this Presentation is About

inheritance in hierarchies of structures

- Hierarchies of structures are important to organize formalizations
- They are implemented with type classes or unification hints
- Problems for developers: performance/predictability,
  not all issues are properly documented
- In this talk, we discuss problems caused by competing definitions
- In particular, we identify **forgetful inheritance** as a solution to
  the problem of competing inheritance paths for poorer structures

Motivation for this Work

analysis with the Coq proof assistant

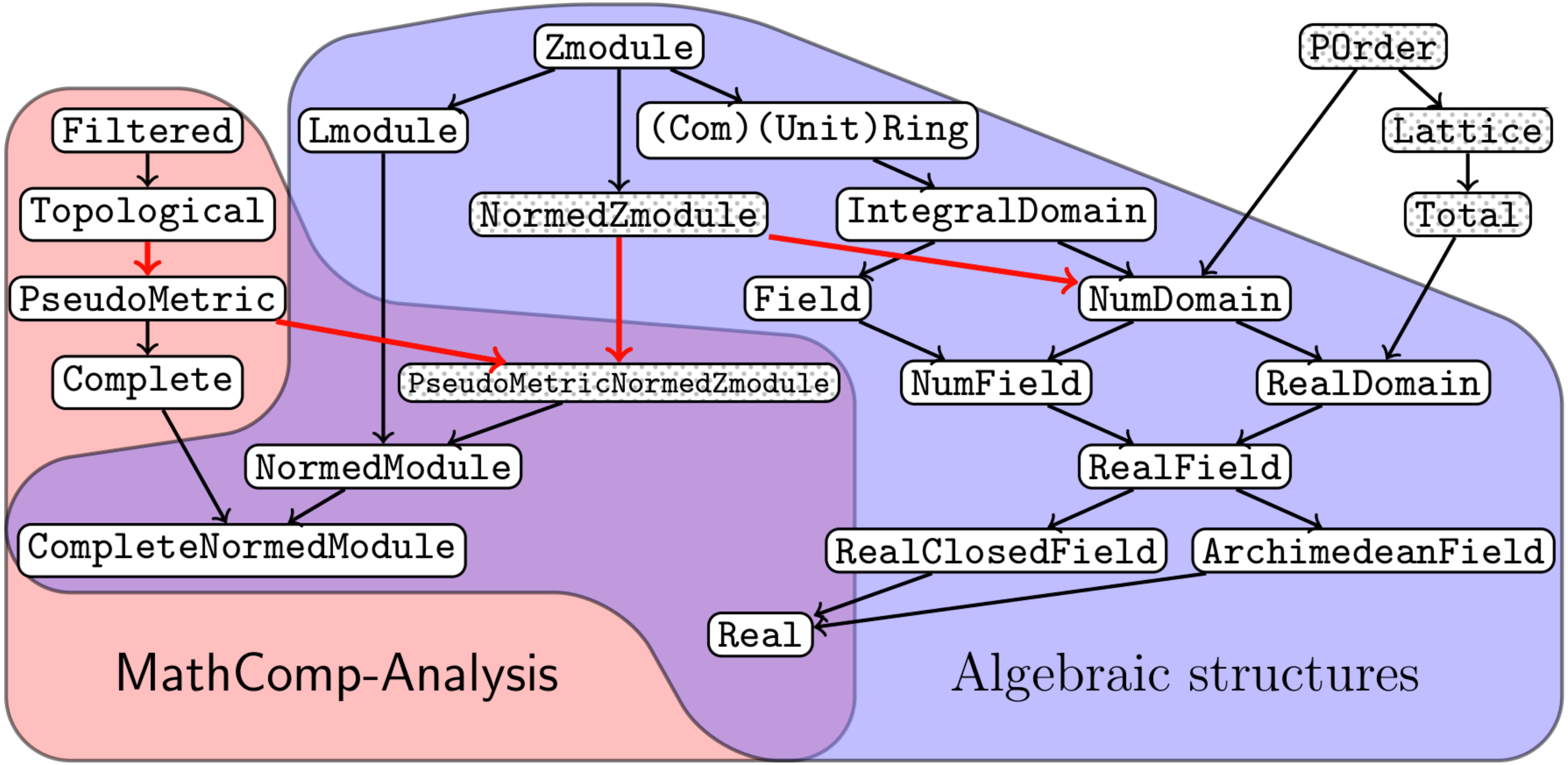
- MathComp is the library for algebra used to formalize the odd order theorem by Gonthier et al. [ITP 2013]

- Coquelicot is a library for real analysis by Boldo et al. [MCS 2015]
  - It extends and improves the standard library of Coq

- MathComp-Analysis is a work-in-progress that extends MathComp for functional analysis.
  - It is inspired by Coquelicot but comes with original features [JFR 2018]

MathComp-Analysis Hierarchy

the concrete result of this presentation (teaser)



Competing Inheritance Paths in Dependent Type Theory

outline

1. Background: Hierarchies of Structures
2. Problem: Extend MathComp with Analysis
3. Forgetful Inheritance for Competing Inheritance Paths
4. Application to MathComp-Analysis
5. Conclusion

$\boxed{\text{module}}$

Mathematical Structure

using a packed class [Garillot et al., 2009]

1️⃣ Record **isModule** T := IsModule {**minus_op** : T -> T -> T}. } Mixin = operator + properties

2️⃣ Structure **module** := Module {

module_carrier : Type ;

module_isModule : isModule module_carrier }.

} Structure = carrier + mixin

3️⃣ Definition **minus** (M : module)

: module_carrier M -> module_carrier M -> module_carrier M

:= minus_op _ (module_isModule M).

} operator lifted from the mixin to the structure

4️⃣ Notation “x - y” := (minus _ x y). } Notation with a hole ≈ overloading

Structure Inference
with unification hints

module □−□ $\xrightarrow{\text{canonical}}$ Z_module

1️⃣ Definition Z_module := Module Z (IsModule Z Z.sub). } structure instance

2️⃣ Canonical Z_module. } the instance is canonical

Check forall x y : Z, x - y = x - y. $\xrightarrow[\text{Notations}]{\text{Unset Printing}}$ minus ? x y

Z

module

module_carrier ?

type equation

? ≡ Z_module $\xleftarrow{\text{solution}}$ module_carrier ? ≡ Z

Inheritance of Structures

using the notion of class

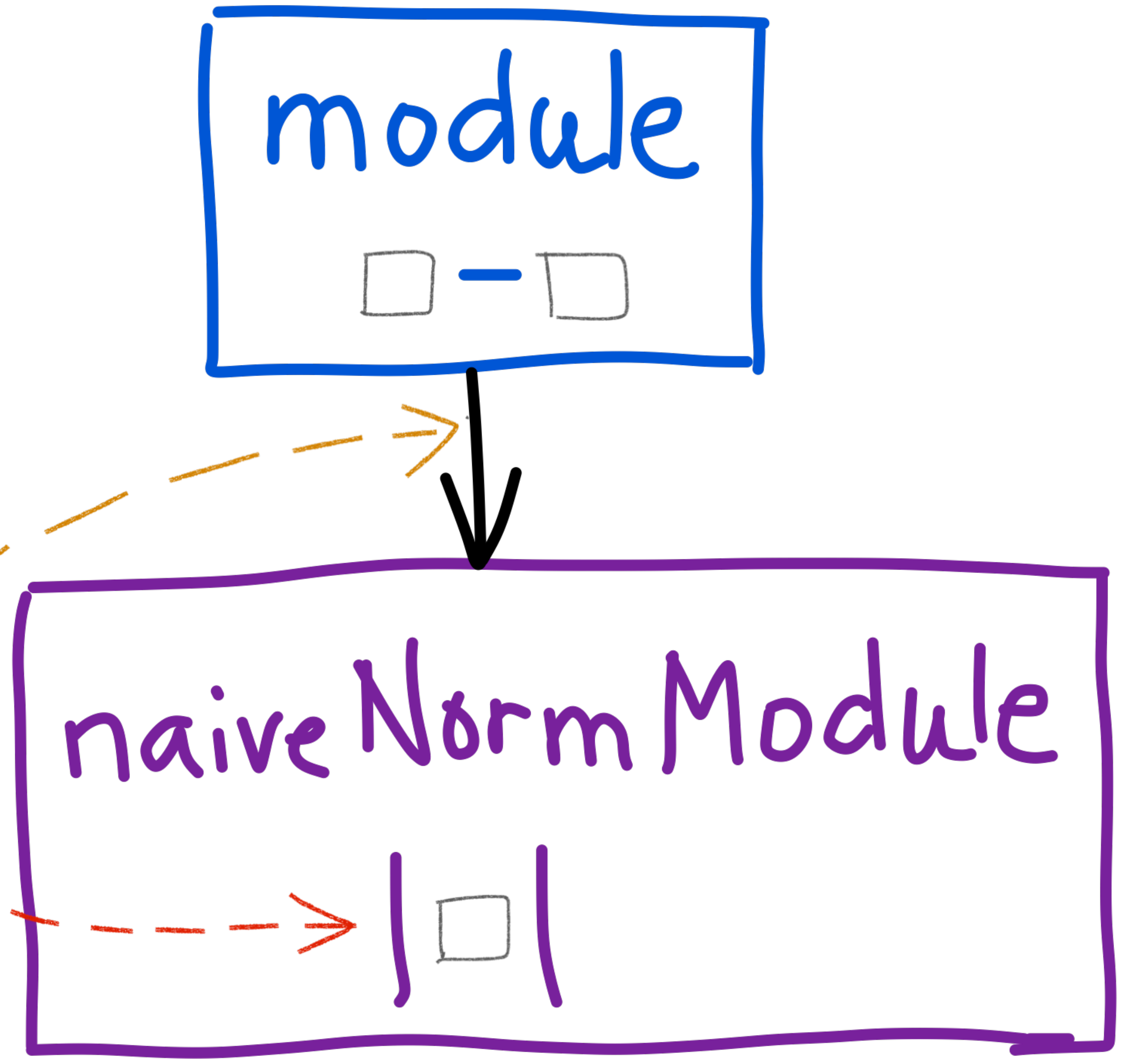
1️⃣ Record **naiveNormMixin** (T : module) := NaiveNormMixin { mixin

naive_norm_op : T -> nat }.

Record **isNaiveNormModule** (T : Type) := IsNaiveNormModule {

nbase : isModule T ; class ≈ inheritance structure

nmix : naiveNormMixin (Module _ nbase) }.



2️⃣ Structure **naiveNormModule** := NaiveNormModule { structure

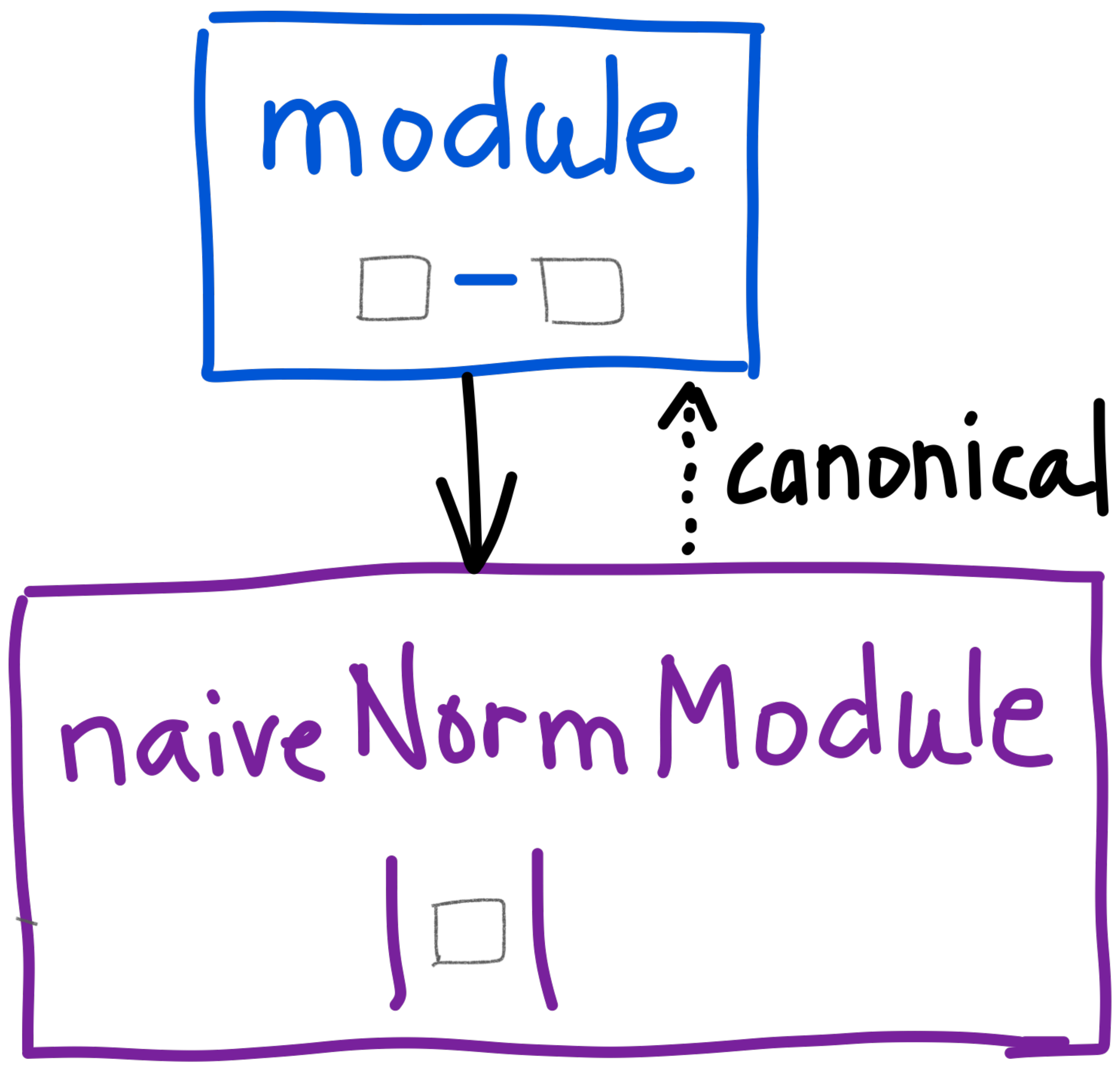
naive_norm_carrier :> Type ;

implicit coercion

naive_normModule_isNormModule : isNaiveNormModule naive_norm_carrier }.

Inference in Presence of Inheritance

using unification hints



Check forall (N : naiveNormModule) (x y : N),

module_carrier ?

x - y = x - y.

implicit coercion

naive_norm_carrier N

KO

Canonical **naiveNorm_isModule** (N : naiveNormModule) :=

Module N (nbase _ (naive_normModule_isNormModule N)).

module_carrier ? ≡ naive_norm_carrier N

OK

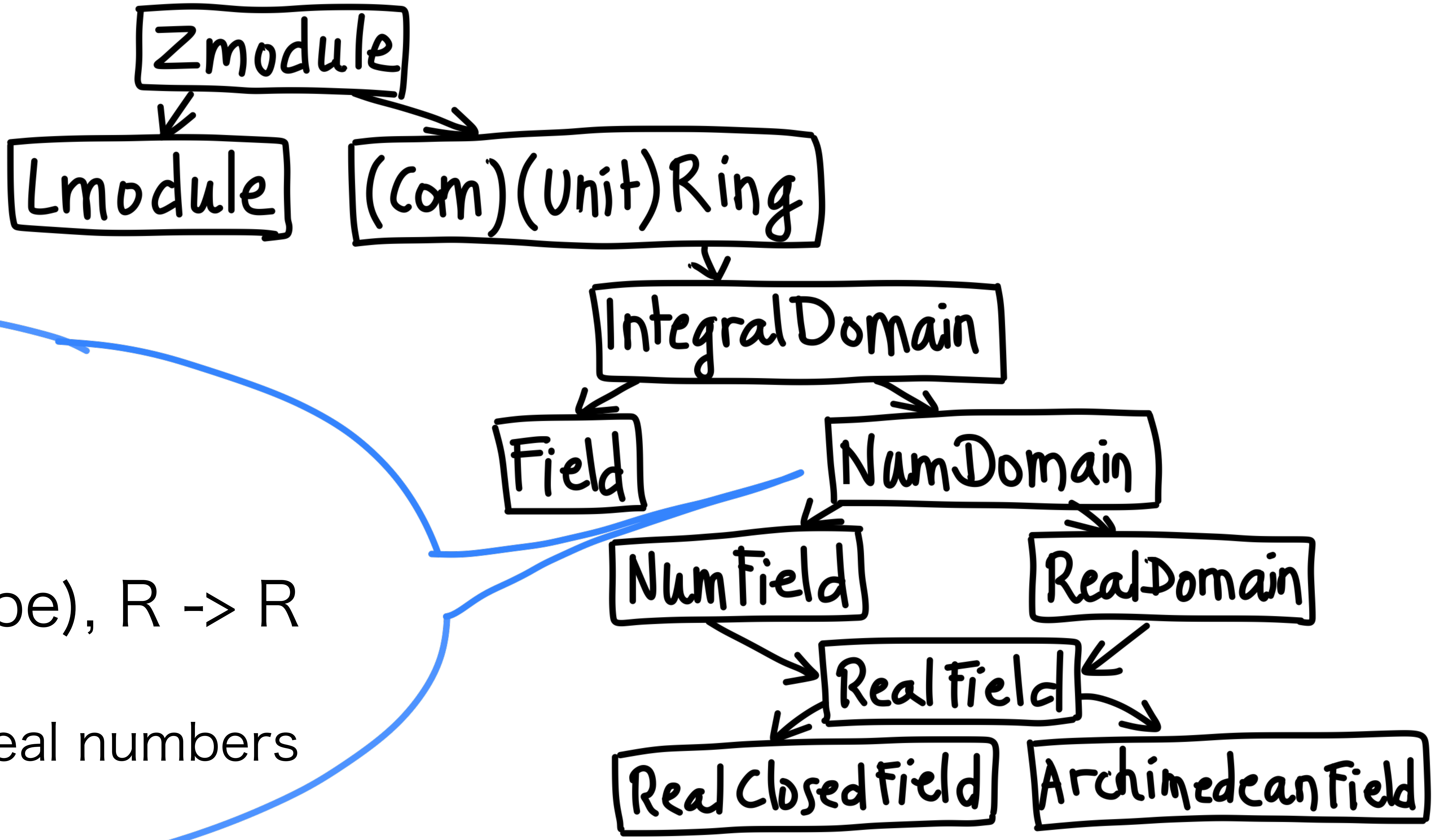
naiveNorm_isModule N

Competing Inheritance Paths in Dependent Type Theory

outline

1. Background: Hierarchies of Structures
2. Problem: Extend MathComp with Analysis
3. Forgetful Inheritance for Competing Inheritance Paths
4. Application to MathComp-Analysis
5. Conclusion

The MathComp Library
(excerpt, before this work)



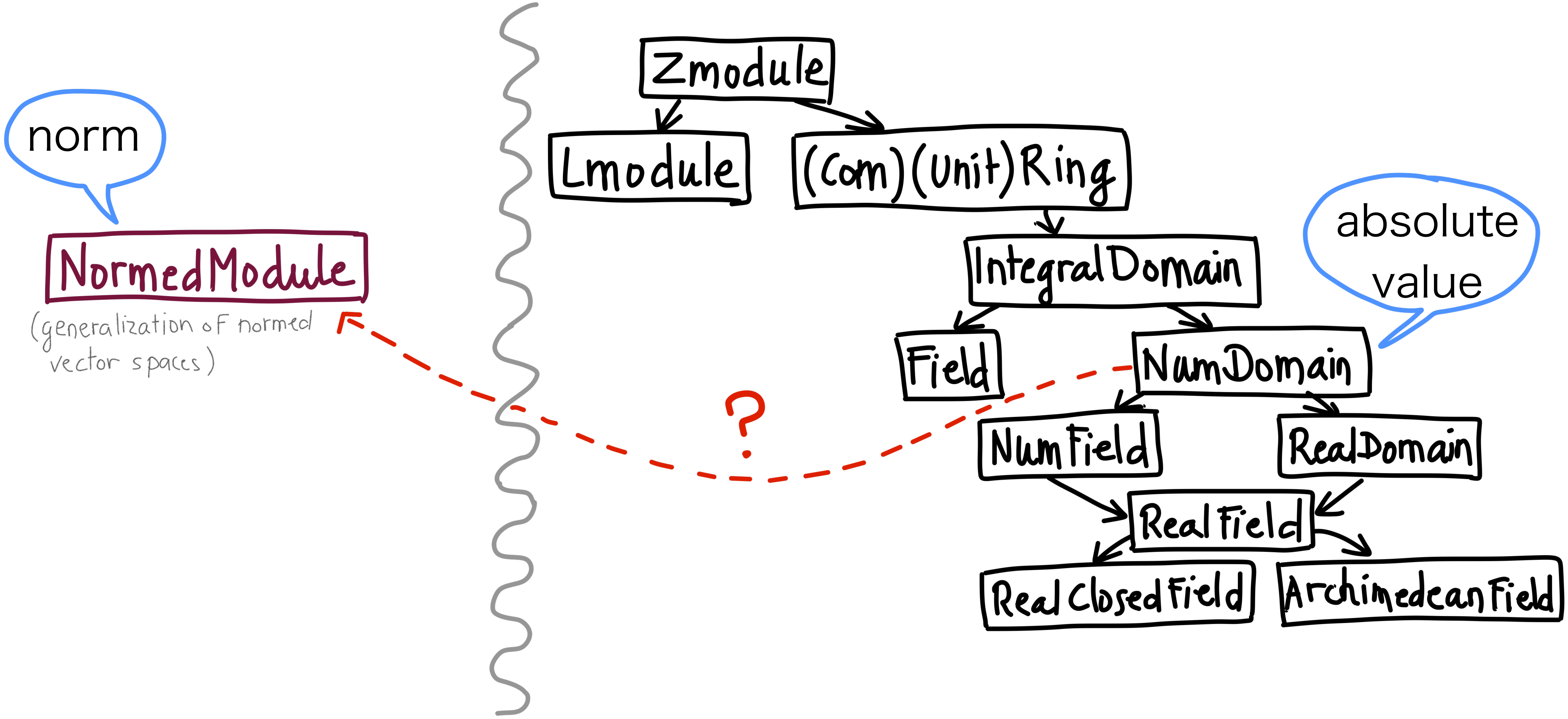
absolute value of type:

forall (R : numDomainType), R -> R

Sample instances: integers, real numbers

Towards Analysis with MathComp

issue #1: properties of differentiability, continuity



Towards Analysis with MathComp

issue #2: topology

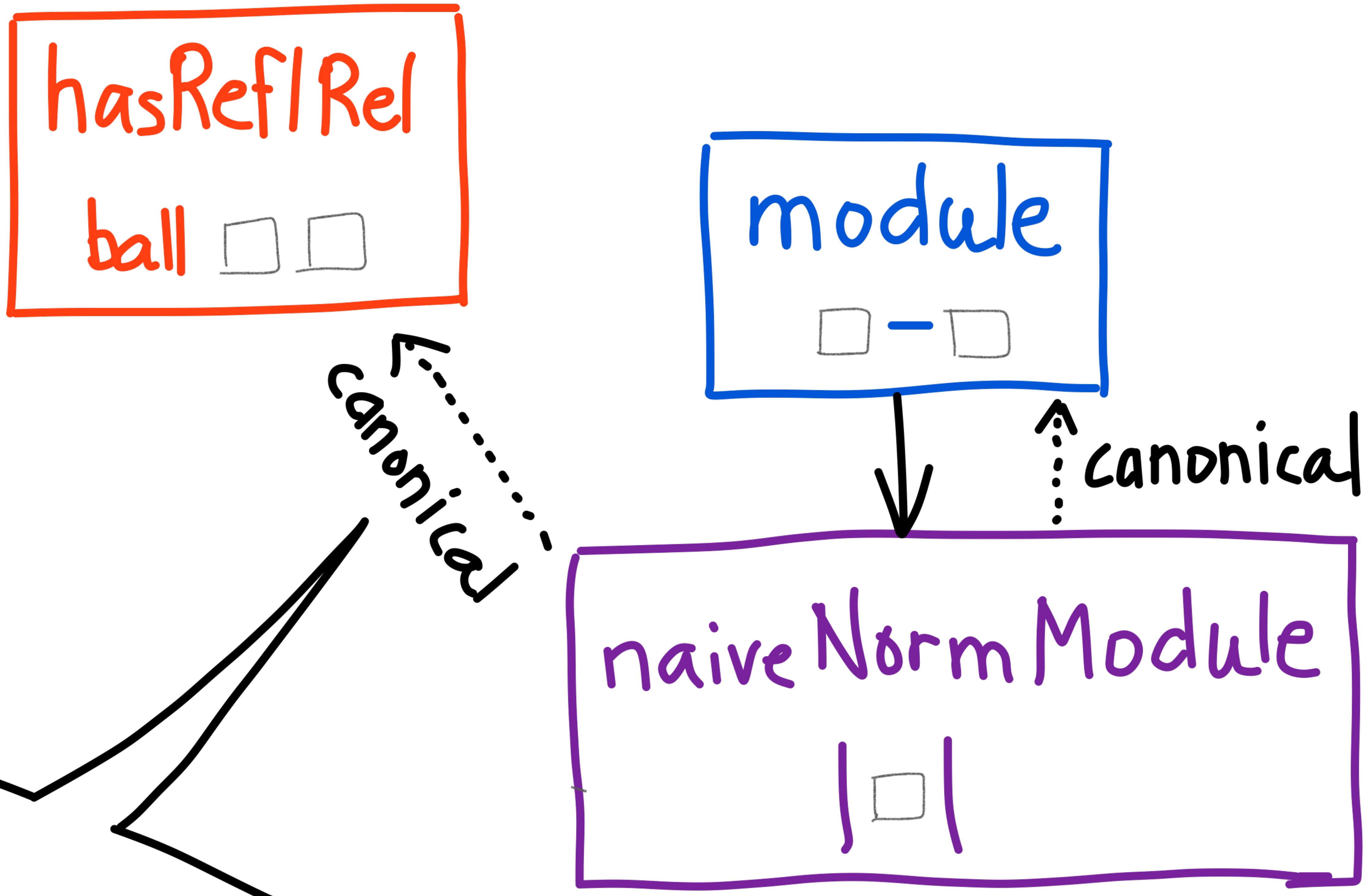
Competing Inheritance Paths in Dependent Type Theory

outline

1. Background: Hierarchies of Structures
2. Problem: Extend MathComp with Analysis
3. Forgetful Inheritance for Competing Inheritance Paths
4. Application to MathComp-Analysis
5. Conclusion

Hierarchy Extension

a new structure with a notion of ball



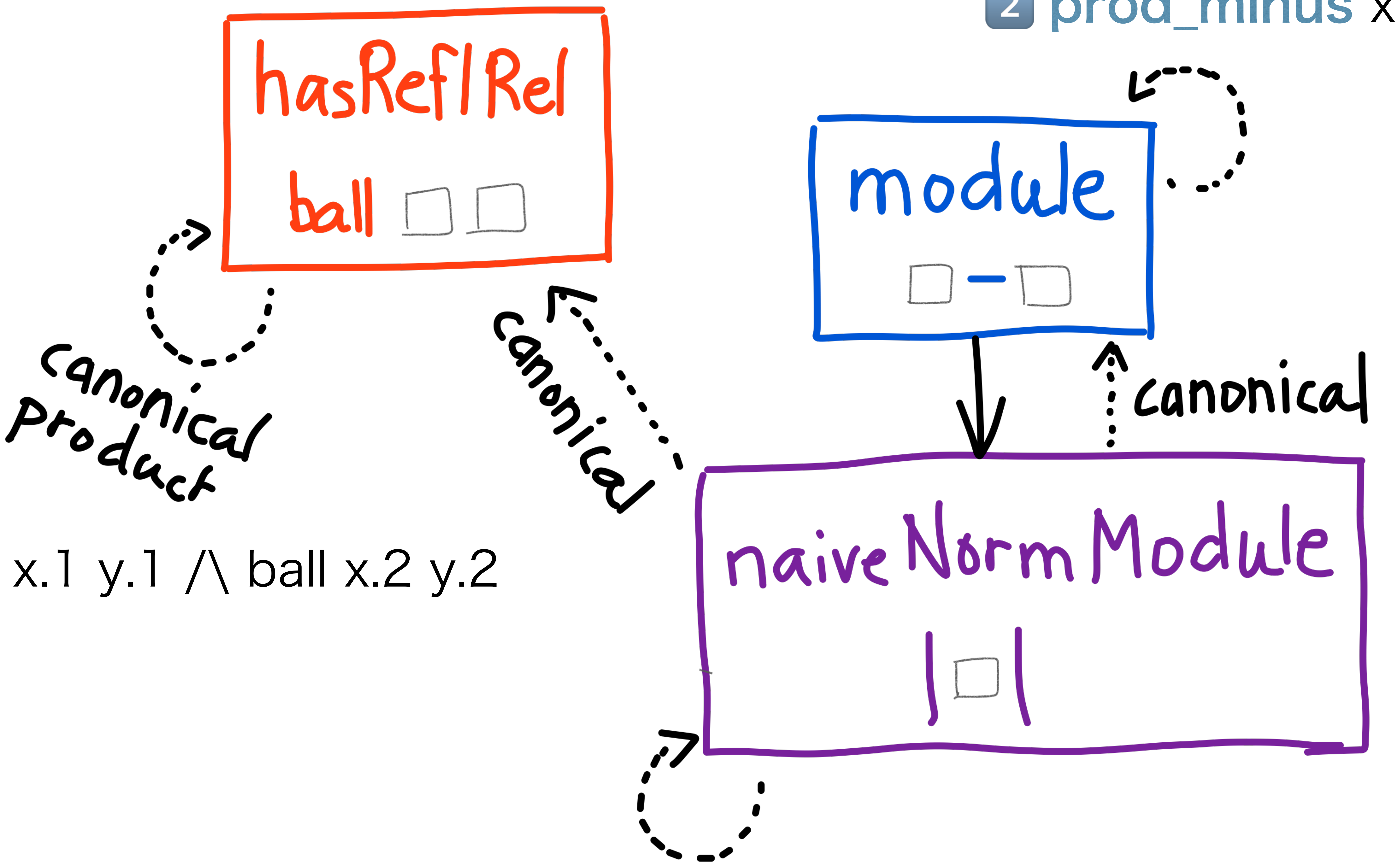
Notion of ball defined using norm

```
Variable N : naiveNormModule.

Definition norm_ball (x : N) := fun y : N => |x - y| ≤ 1.

Definition naiveNormModule_isReflRel := IsReflRel N norm_ball … .

Canonical nnorm_hasReflRel := HasReflRel N naiveNormModule_isReflRel
```

Hierarchy Extension

canonical products for all structures

2️⃣ prod_minus x y := (x.1 - y.1, x.2 - y.2).



1️⃣ prod_ball x y := ball x.1 y.1 /\ ball x.2 y.2

3️⃣ prod_nnorm x := max |x.1| |x.2|

A Puzzling Inference Failure

Example failure

Hypothesis → (Pball : forall V : naiveNormModule, forall v : V, **P** (ball v))

(W : naiveNormModule) (w : W * W)

Goal → : **P** (ball w).

Proof. Fail apply Pball. Abort.

Shouldn't this succeed since there is a canonical way to build products? 🤔

A Puzzling Inference Failure

prod_naiveNormModule W W

Example failure   nnorm_hasReflRel ?

(Pball : forall V : naiveNormModule, forall v : V, **P** (ball v))

(W : naiveNormModule) (w : W * W)

: **P** (ball w).

Proof. Fail apply Pball. Abort.

prod_hasReflRel (nnorm_hasReflRel W)
(nnorm_hasReflRel W)

A Puzzling Inference Failure

Example failure

(Pball : forall V : naiveNormModule, forall v : V, **P** (ball v))

(W : naiveNormModule) (w : W * W)

: **P** (ball w).

Proof. Fail apply Pball. Abort.

prod_naiveNormModule W W

nnorm_hasReflRel ?

$\text{ball\_op}\ x\ y \triangleq |x-y| \leq 1$

$\not\equiv$ KO

not definitionally equal

$\text{ball\_op}\ x\ y \triangleq \text{ball}\ x_1\ y_1 \wedge \text{ball}\ x_2\ y_2$

prod_hasReflRel (nnorm_hasReflRel W) (nnorm_hasReflRel W)

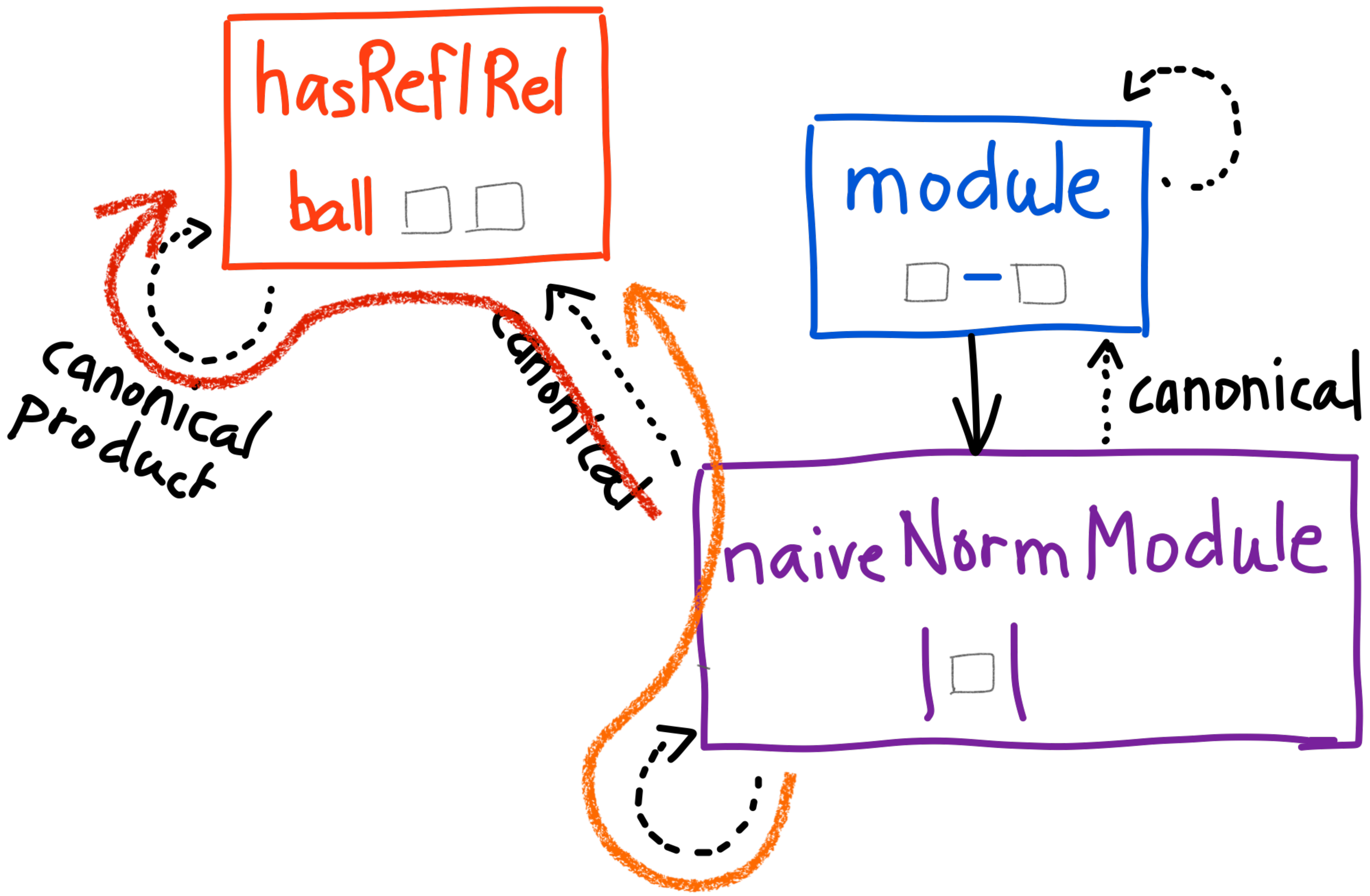
Problem of Competing Inheritance Paths

diagrammatically

(hypothesis in the example)

$$\begin{tikzcd}
\texttt{naiveNormModule} * \texttt{naiveNormModule} \arrow[r, "\texttt{prod_naiveNormModule}"] \arrow[d, "\texttt{nnorm_hasReflRel}"'] \arrow[dr, phantom, "\not\circlearrowleft"] & \texttt{naiveNormModule} \arrow[d, "\texttt{nnorm_hasReflRel}"] \\
\texttt{hasReflRel} * \texttt{hasReflRel} \arrow[r, "\texttt{prod_hasReflRel}"'] & \texttt{hasReflRel}
\end{tikzcd}$$

(goal in the example)



Forgetful Inheritance

how to fix the hierarchy

has Refl Rel as a parameter

1 Record **normMixin** (T : module) (m : isReflRel) := NormMixin {

norm_op : T -> nat ;

compatibility norm ⇕ ball

norm_ball_opP : forall x y, ball_op _ m x y <-> norm_op (x - y) ≤ 1 }.

2 Record **isNormModule** (T : Type) := IsNormModule {

base : isModule T ;

bmix : isReflRel T ;

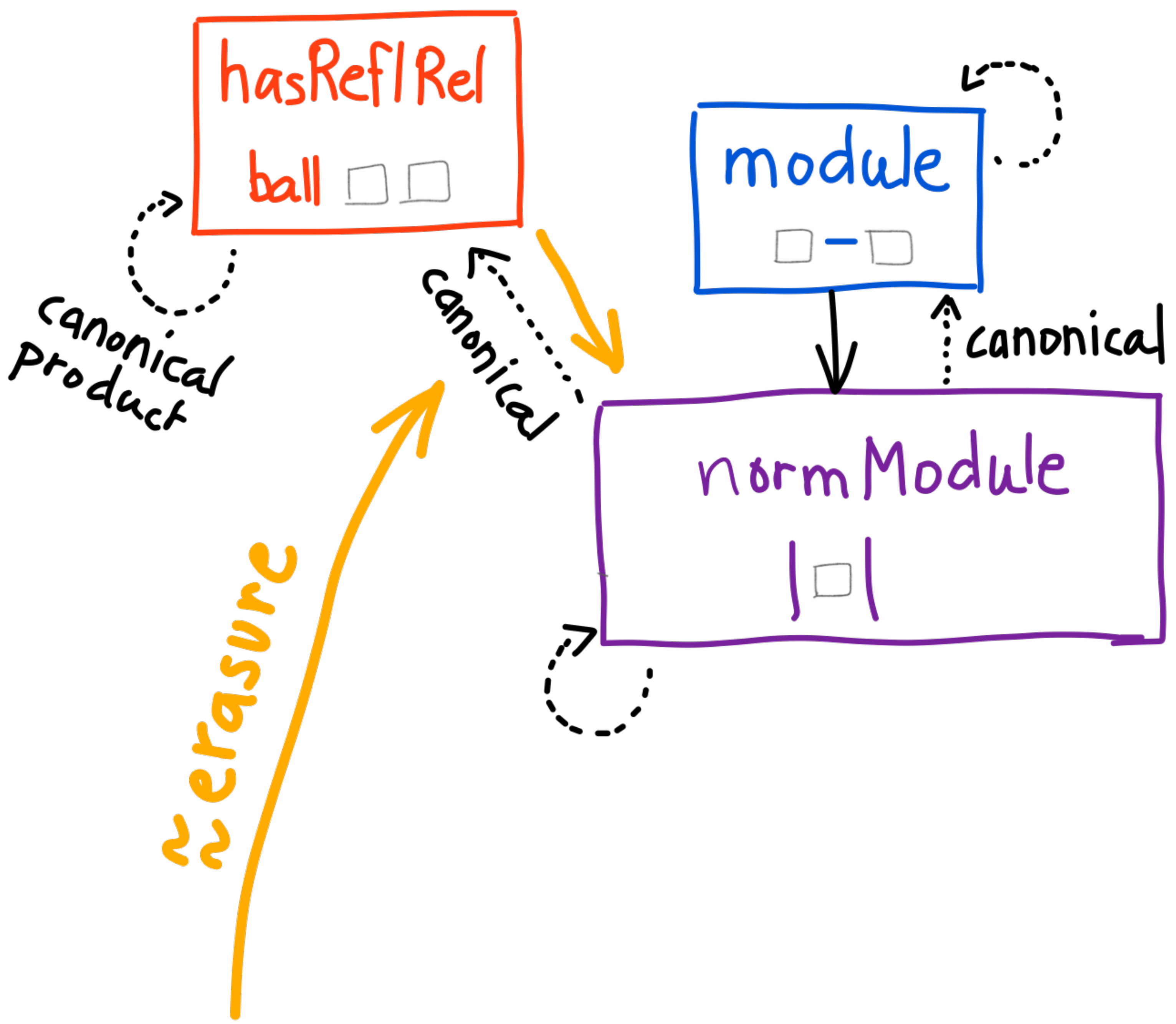
has Refl Rel mixin included in the class

mix : normMixin (Module _ base) bmix }.

3 Structure **normModule** := NormModule {

norm_carrier :> Type ;

normModule_isNormModule : isNormModule norm_carrier }.



4 Canonical **norm_hasReflRel** (N : normModule) :=

HasReflRel N (bmix _ (normModule_isNormModule N))

Competing Inheritance Paths with Type Classes

same problem, same solution

Type classes in Lean

Type classes in Coq

Packed classes in Coq

```
-------------------------------
-- Section 2.2               --
-- isModule is in fact has_sub --
-------------------------------
----------------
-- Section 2.3 --
----------------
universe u
```

```
From Coq Require Import ssreflect ZArith.
Reserved Notation "| x |"   (at level 30).
Reserved Notation  "x ~~ y" (at level 30).
```

```
From Coq Require Import ssreflect ZArith.
Reserved Notation "| x |"   (at level 30).
Reserved Notation  "x ~~ y" (at level 30).

(**************)
(* Section 2.1 *)
(**************)
Record flatNormSpace := FlatNormSpace {
  carrier : Type ;
  fminus : carrier -> carrier -> carrier;
  fnorm : carrier -> nat;
  fnormP : forall x : carrier, fnorm (fminus x x) = 0}.

Check fminus.
(* fminus : forall f : flatNormSpace,                    *)
(*   carrier f -> carrier f -> carrier f        *)
Check fnormP.
(* fnormP : forall (f : flatNormSpace) (x : carrier f), *)
(*   fnorm f (fminus f x x) = 0                        *)

Lemma Z_normP (n : Z) : Z.abs_nat (Z.sub n n) = 0.
Proof. by rewrite Z.sub_diag. Qed.
Definition Z_flatNormSpace := FlatNormSpace Z Z.sub Z.abs_nat Z_normP.

(**************)
(* Section 2.2 *)
(**************)
Record isModule T := IsModule {minus_op : T -> T -> T}.
Structure module := Module {
  module_carrier : Type ;
  module_isModule : isModule module_carrier}.

Definition minus {M : module} := minus_op _ (module_isModule M).
Notation "x - y" := (minus x y).

Definition Z_isModule : isModule Z := IsModule Z Z.sub.
Definition Z_module := Module Z Z_isModule.

Fail Check forall x y : Z, x - y = x - y.

Canonical Z_module.
Check forall x y : Z, x - y = x - y.
Print Canonical Projections.
(* the line Z <- module_carrier (Z_module) tries to solve *)
(* the equation `module_carrier ?M =?= Z`                *)

(* begin omitted from the paper *)
Coercion module_carrier : module >-> Sortclass.
(* end omitted from the paper   *)
```

https://math-comp.github.io/competing-inheritance-paths-in-dependent-type-theory/

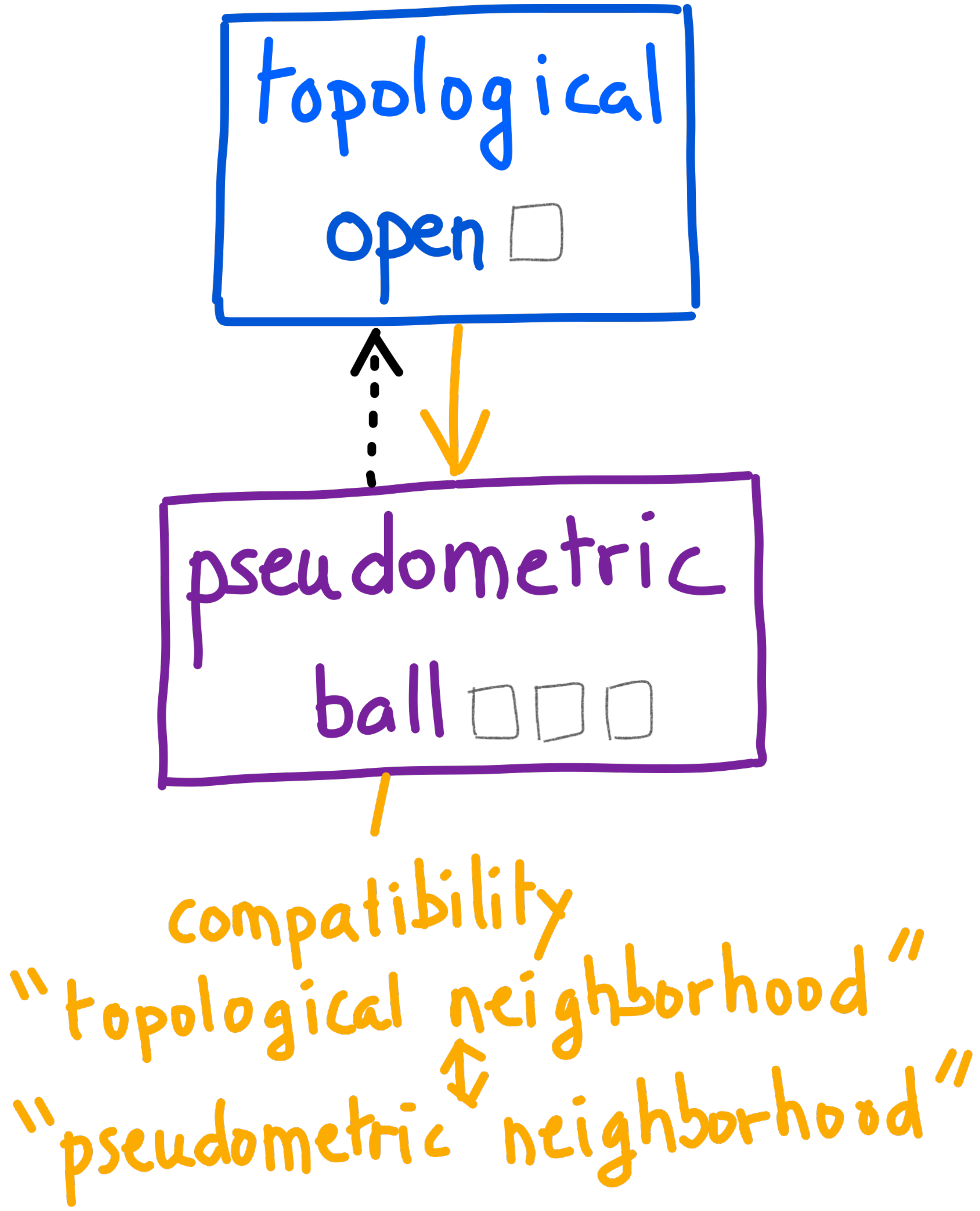
Competing Inheritance Paths in Dependent Type Theory

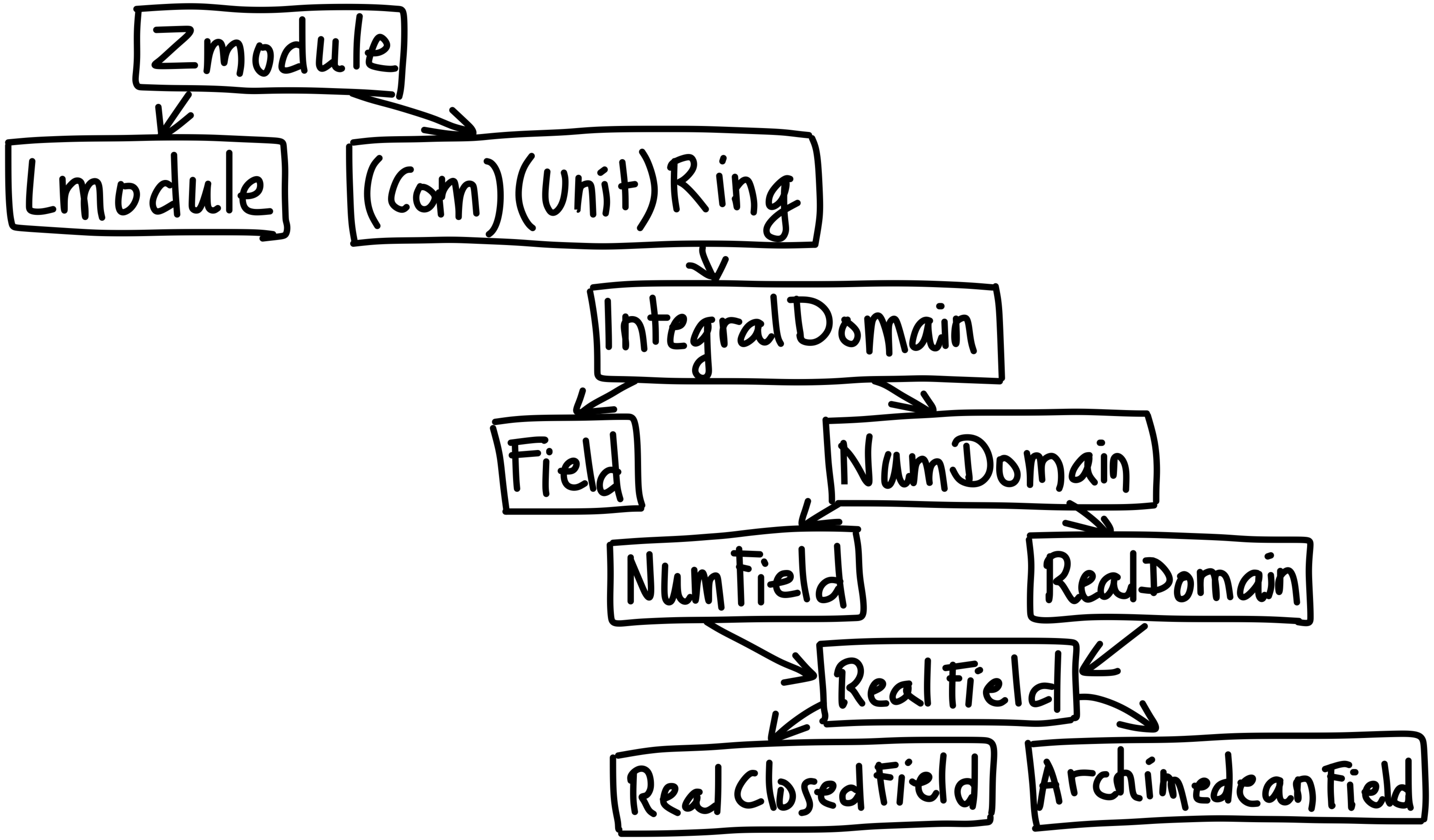
outline

1. Background: Hierarchies of Structures
2. Problem: Extend MathComp with Analysis
3. Forgetful Inheritance for Competing Inheritance Paths
4. Application to MathComp-Analysis
5. Conclusion

Forgetful Inheritance

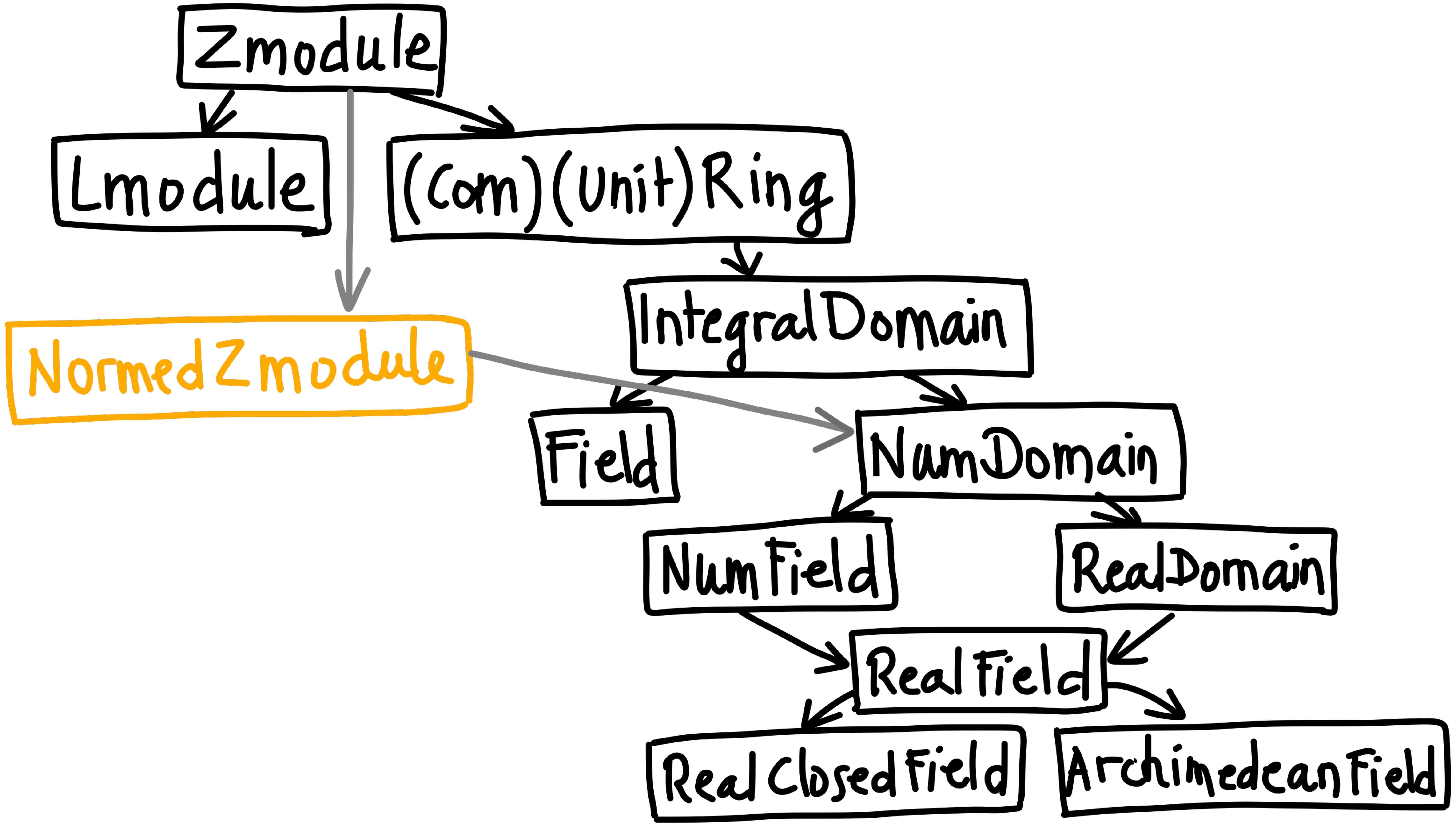
from pseudometric spaces to topological spaces





Unify Absolute Value and Norm

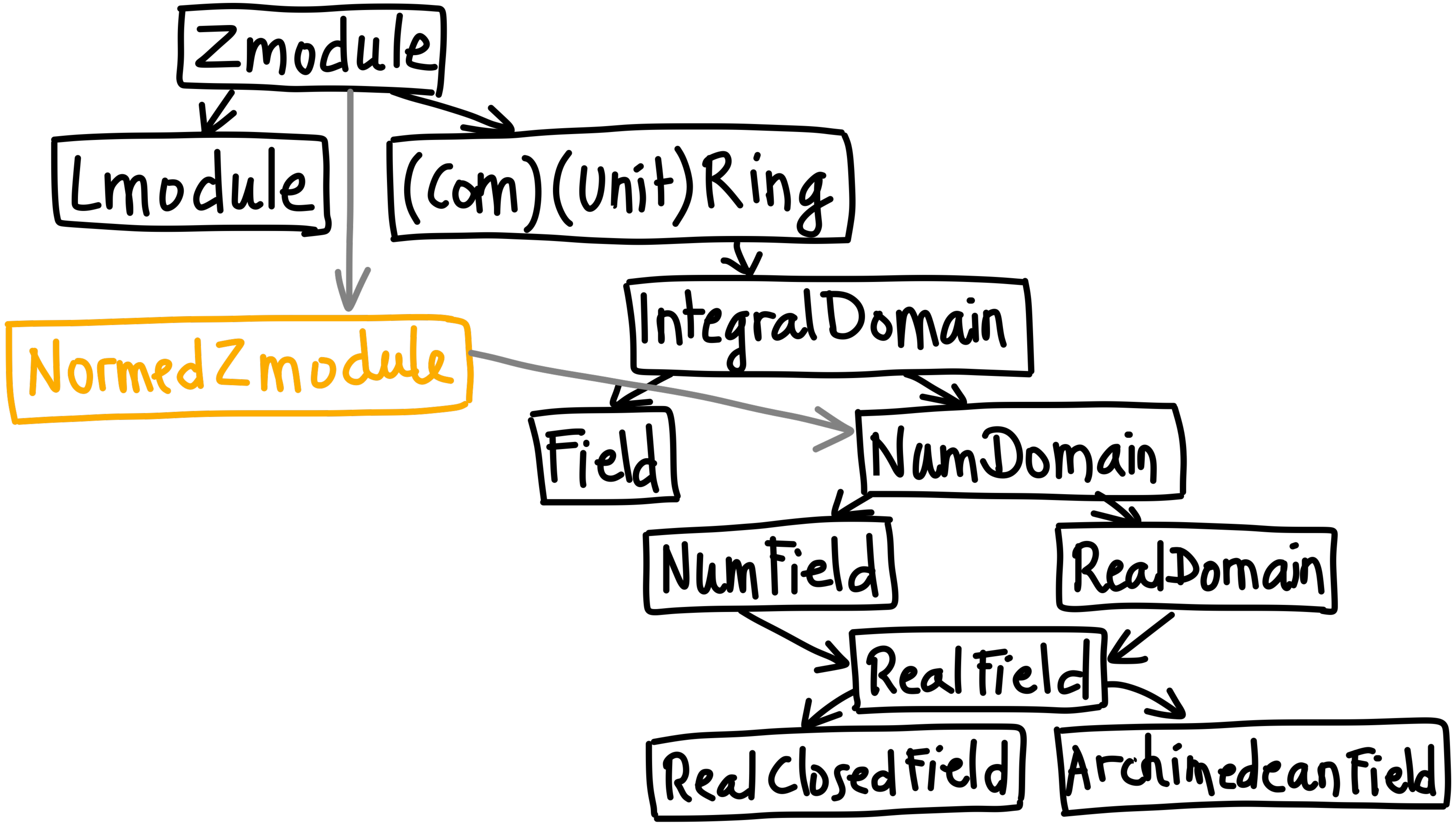
step #1: factorize with normed Abelian groups



Unify Absolute Value and Norm

step #2: solve mutual dependency problem

properties of the norm require the codomain of the norm to be a normed Abelian group and an order (e.g., triangle inequality)... 🤔

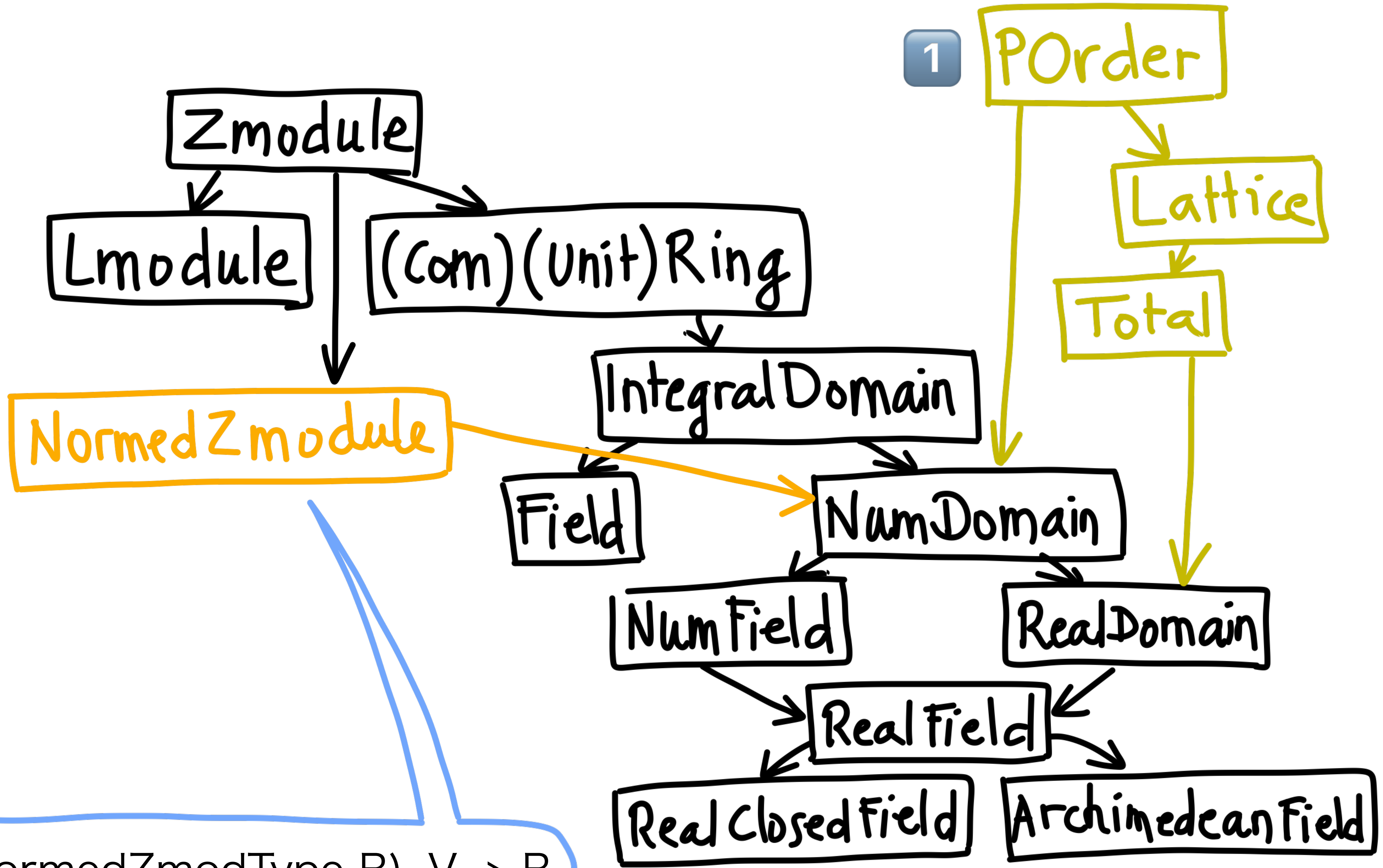


Forgetful Inheritance
from numerical domain to normed Abelian group

2️⃣ the class of NumDomain changed to include the mixin of NormedZmodule

3️⃣ Forgetful inheritance:

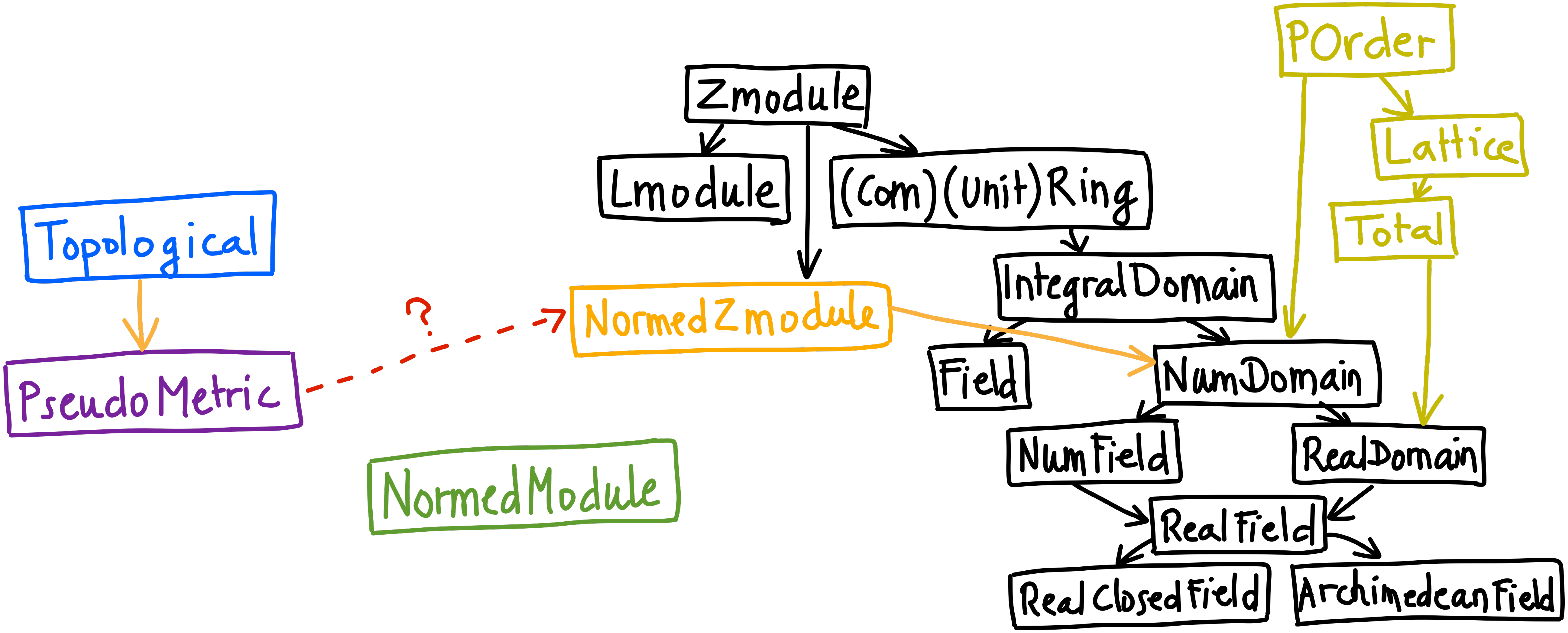
1. the class of NormedZmodule is parameterized by NumDomain and features the mixin of the norm
2. inheritance from NormedZmodule to NumDomain is declared



norm : forall (R : numDomainType) (V : normedZmodType R), V -> R

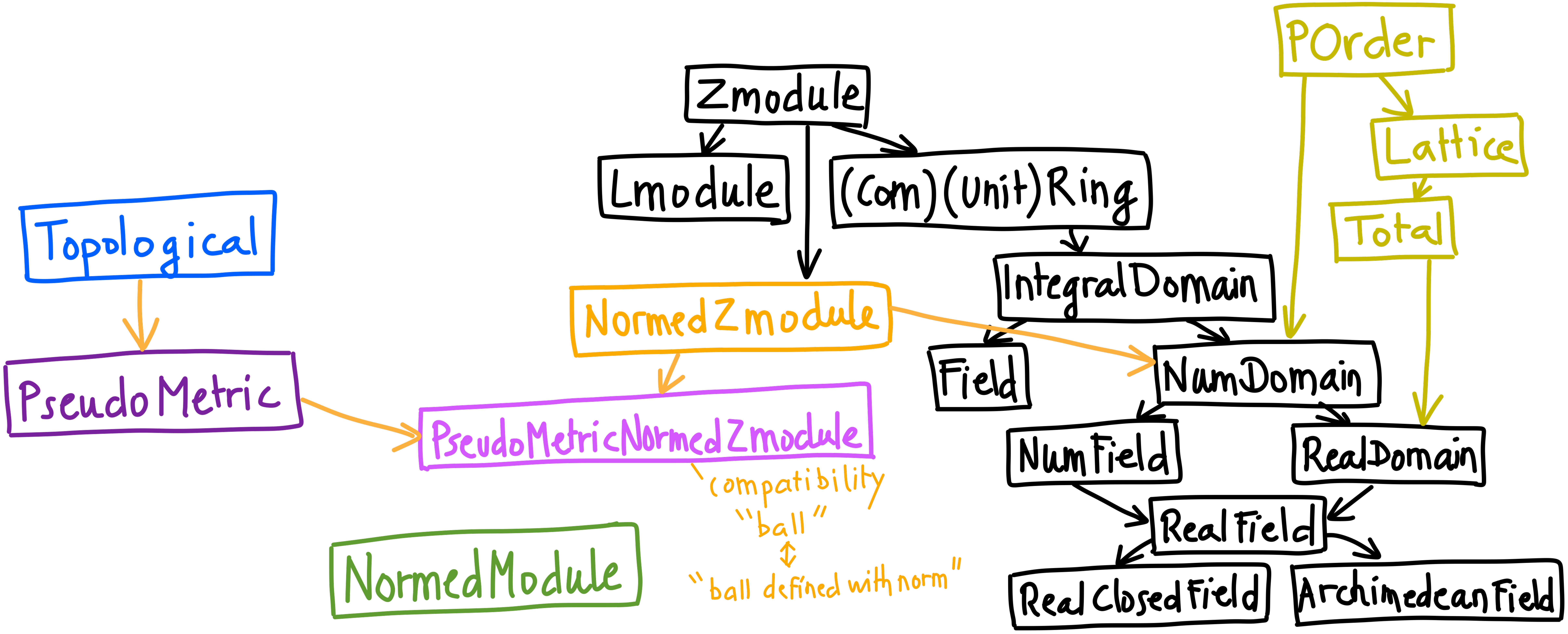
How do we Integrate Normed Modules?

final step



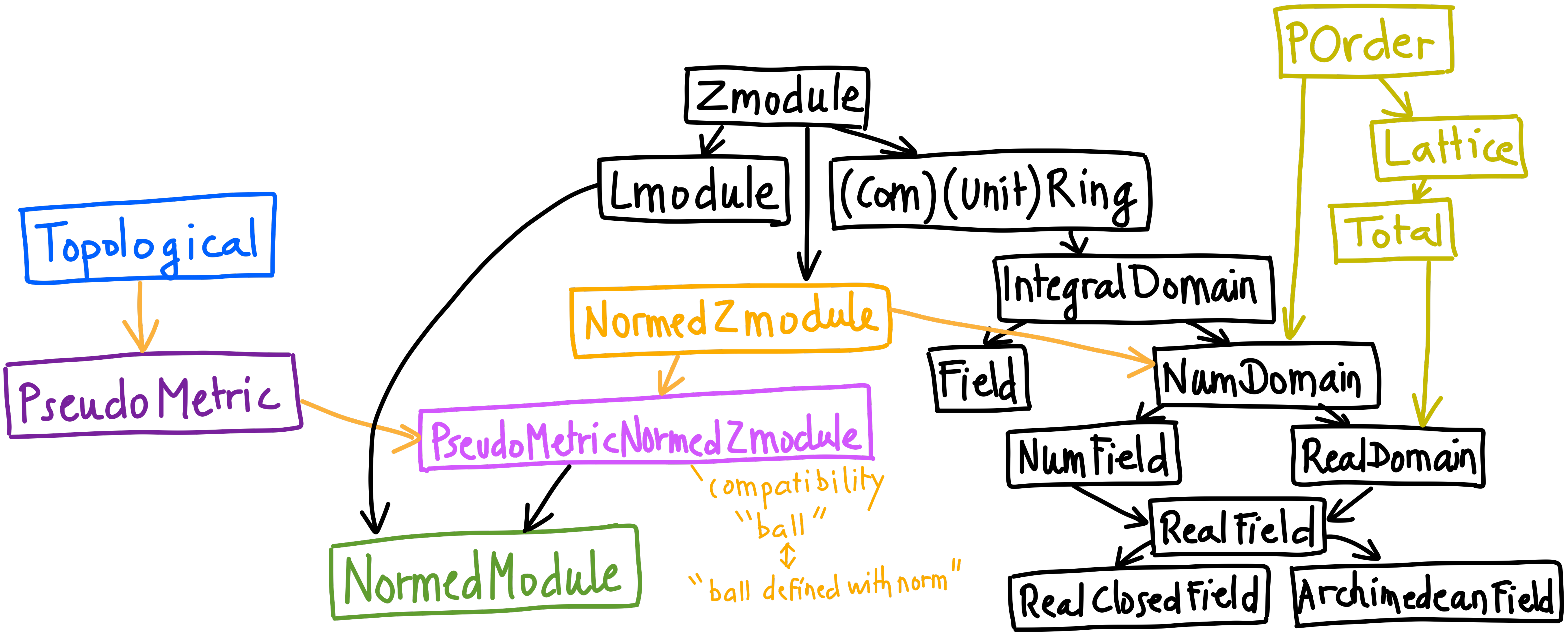
Forgetful Inheritance

from normed modules to pseudometric spaces



Forgetful Inheritance

from normed modules to pseudometric spaces



Competing Inheritance Paths in Dependent Type Theory

outline

1. Background: Hierarchies of Structures
2. Problem: Extend MathComp with Analysis
3. Forgetful Inheritance for Competing Inheritance Paths
4. Application to MathComp-Analysis
5. Conclusion

Related Work

- HOL Light and Isabelle/HOL have more real and complex analysis
  - different foundations: we use dependent types here
- Forgetful inheritance was already mentioned
  - original description of packed classes [TPHOLs 2009] alludes to it
  - Buzzard et al. [CPP 2020] discuss a similar issue, on a specific example
- Packed classes + forgetful inheritance is verbose
  - Sakaguchi [IJCAR 2020] developed an automated checking tool ← on Friday
  - Cohen et al. [FSCD 2020] are working on automated generation ← tomorrow

Conclusion

summary of main contributions

1️⃣ forgetful inheritance using packed classes

(several examples, comparison with type classes)

2️⃣ the MathComp-Analysis hierarchy

(enhancement of MathComp,

applications of forgetful inheritance)

Available: theories of Bachmann-Landau notation, of differentiability