Univalent foundations and the equivalence principle

Paige Randall North

16 October 2020

Outline

The equivalence principle

Equivalence principle

Reasoning in mathematics should be **invariant under** the appropriate notion of **equivalence**.

The equivalence principle

Equivalence principle

Reasoning in mathematics should be **invariant under** the appropriate notion of **equivalence**.

Notion of equivalence depends on the objects under consideration:

- **equal** numbers, functions,. . .
- **isomorphic** sets, groups, rings,. . .
- **equivalent** categories
- **biequivalent** bicategories
- . . .

Non-examples: statements violating equivalence principle

We can easily **violate** this principle:

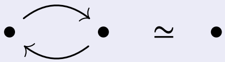
Exercise

Find a statement about sets that is not invariant under isomorphism:

$$\{\emptyset, \{\emptyset\}\} \cong \{\emptyset, \{\{\emptyset\}\}\}$$

Exercise

Find a statement about categories that is not invariant under equivalence:



Non-examples: statements violating equivalence principle

We can easily **violate** this principle:

Exercise

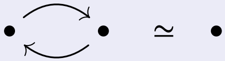
Find a statement about sets that is not invariant under isomorphism:

$$\{\emptyset, \{\emptyset\}\} \cong \{\emptyset, \{\{\emptyset\}\}\}$$

$\{\emptyset\} \in X$

Exercise

Find a statement about categories that is not invariant under equivalence:



$\mathscr{C}$ has exactly 1 object.

A language for invariant properties

Michael Makkai, *Towards a Categorical Foundation of Mathematics*:

> *"The basic character of the Principle of Isomorphism is that of a* ***constraint on the language*** *of Abstract Mathematics; a welcome one, since it provides for the separation of sense from nonsense."*

A language for invariant properties

Michael Makkai, *Towards a Categorical Foundation of Mathematics*:

> *"The basic character of the Principle of Isomorphism is that of a* ***constraint on the language*** *of Abstract Mathematics; a welcome one, since it provides for the separation of sense from nonsense."*

Goal

To have a **syntactic criterion** for properties and constructions that are invariant under equivalence

How to break the equivalence principle for categories. . .

- Recall: the statement

  The category $\mathscr{C}$ has exactly one object.

  is not invariant under equivalence of categories.
- In general, referring to **equality of objects** breaks invariance, but. . .

How to break the equivalence principle for categories. . .

- Recall: the statement

  The category $\mathscr{C}$ has exactly one object.

  is not invariant under equivalence of categories.
- In general, referring to **equality of objects** breaks invariance, but. . .
- even the **definition** of category refers to equality of objects:

Problem

"If $\mathrm{dom}(g)$ is **equal to** $\mathrm{cod}(f)$, then $g \circ f$ exists."

How to break the equivalence principle for categories. . .

- Recall: the statement

  The category $\mathscr{C}$ has exactly one object.

  is not invariant under equivalence of categories.
- In general, referring to **equality of objects** breaks invariance, but. . .
- even the **definition** of category refers to equality of objects:

Problem

"If $\mathrm{dom}(g)$ is **equal to** $\mathrm{cod}(f)$, then $g \circ f$ exists."

Can we give a definition of category without using equality of objects?

. . . and how to fix it.

Solution

Use a logic/language of **dependent sets**, in which $\mathrm{dom}(g) = \mathrm{cod}(f)$ is encoded by what type of thing f and g are.

. . . and how to fix it.

Solution

Use a logic/language of **dependent sets**, in which $\mathrm{dom}(g) = \mathrm{cod}(f)$ is encoded by what type of thing f and g are.

A category consists of

- a set O of objects
- for each $x, y \in O$, a type/set $A(x, y)$ of arrows
- for each $x, y, z \in O$ and each $f \in A(x, y)$ and $g \in A(y, z)$, a type/set $g \circ f \in A(x, z)$
- for each $x \in O$, an identity $\mathrm{id}_x \in A(x, x)$
- . . .

. . . and how to fix it.

Solution

Use a logic/language of **dependent sets**, in which $\text{dom}(g) = \text{cod}(f)$ is encoded by what type of thing f and g are.

A category consists of

- a set O of objects
- for each $x, y \in O$, a type/set $A(x, y)$ of arrows
- for each $x, y, z \in O$ and each $f \in A(x, y)$ and $g \in A(y, z)$, a type/set $g \circ f \in A(x, z)$
- for each $x \in O$, an identity $\text{id}_x \in A(x, x)$
- . . .

Gives rise to **dependently typed language** by adding logical connectors.

Invariance for statements

Theorem (Freyd '76, Blanc '78)

A property of categories (expressed in 2-sorted first order logic) is invariant under equivalence iff it can be expressed in this dependently typed language, using equality for arrows but not for objects.

Invariance for statements

Theorem (Freyd '76, Blanc '78)

A ***property*** *of categories (expressed in 2-sorted first order logic) is invariant under equivalence iff it can be expressed in this dependently typed language, using equality for arrows but not for objects.*

- What about **constructions** on categories?

Invariance for statements

Theorem (Freyd '76, Blanc '78)

A ***property*** *of categories (expressed in 2-sorted first order logic) is invariant under equivalence iff it can be expressed in this dependently typed language, using equality for arrows but not for objects.*

- What about **constructions** on categories?
- What about other mathematical structures?

Outline

Overview of types in Martin-Löf type theory

Type former	Notation	canonical term
Dependent type	$x : A \vdash B(x)$	
Dependent term	$x : A \vdash b(x) : B(x)$	
Boolean type	Bool	$\top, \bot$
Natural numbers type	Nat	$0, sx$
Sum type	$\sum_{x:A} B(x)$	(a, b)
Product type	$\prod_{x:A} B(x)$	$\lambda(x : A).b$
Identity type	$x : A, y : A \vdash x = y$	$\mathrm{refl}(x) : x = x$
Universe	Type	

Curry-Howard Correspondence

We can interpret these types as propositions or sets.

Properties of the identity type

Induction principle for $a = b$

To define a function

$$f : \prod_{(x,y:A)} \prod_{(p:x=y)} C(x,y,p)$$

it suffices to specify its image on $(x, x, \mathrm{refl} x)$.

- sym : $\prod_{x,y:A}(x = y) \to (y = x)$
- trans : $\prod_{x,y,z:A}(x = y) \times (y = z) \to (x = z)$

The equality principle in type theory

Any predicate or construction that can be defined on terms of a type A is of the form $f : A \to B$.

- The predicate "G is an abelian group" is a function $Grp \to Prop$.
- Considering the lattice of subgroups of any group G produces a function $Grp \to Latt$.

Equality principle

$$\prod_{x,y:A} (x = y) \to \prod_{f:A \to B} \big(f(x) = f(y)\big)$$

Space interpretation

The identity type behaves like equality:

- reflexivity, symmetry, transitivity
- Everything respects equality

but more like paths in a space:

- Can iterate identity type
- Cannot show that any two identities are identical

Voevodsky Correspondence

We can interpret

- a type K as a *Kan complex* $[K]$
- a dependent type $x : B \vdash E(b)$ as a *Kan fibration* $[p] : [E] \to [B]$
- a dependent term $x : B \vdash e(b) : E(b)$ as a *section* of $[e]$ of $[p]$
- a term $p : a \to_K b$ as a path from a to b in K

The Univalence Axiom

There are two notions of 'sameness' between types:

- $A = B$
- $A \simeq B$ (functions $f : A \leftrightarrows B : g$ such that $fg = 1$ and $gf = 1$)

There is always a function

$$(A = B) \to (A \simeq B)$$

which is an equivalence in Kan complexes.

The Univalence Axiom

The function

$$(A = B) \to (A \simeq B)$$

is an equivalence.

This is true in Kan complexes.

Outline

Strategy

We always have a version of the equivalence principle:

Equality principle

$$\prod_{x,y:A}(x = y) \to \prod_{f:A\to B}\big(f(x) = f(y)\big)$$

but we want better ones where we replace the 'synthetic' equality $x = y$ with an 'analytic' equality $x \cong y$ which depends on the type.

Strategy: prove that the function $(x = y) \to (x \cong y)$ is an equivalence

Univalence principle

$(x =_T y) \cong (x \cong_T y)$

for a type T and appropriate $\cong_T$. Then we will get:

Equivalence principle

$$\prod_{x,y:A}(x \cong y) \to \prod_{f:A\to B}\big(f(x) = f(y)\big)$$

Contractible types, propositions and sets

- A is **contractible**

$$\mathsf{isContr}(A) \;:\equiv\; \sum_{x:A}\prod_{y:A} y = x$$

- A is a **proposition**

$$\mathsf{isProp}(A) \;:\equiv\; \prod_{x,y:A} x = y$$

- A is a **set**

$$\mathsf{isSet}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isProp}(x = y)$$

$$\mathsf{Prop} \;:\equiv\; \sum_{X:\mathsf{Type}} \mathsf{isProp}(X) \qquad \mathsf{Set} \;:\equiv\; \sum_{X:\mathsf{Type}} \mathsf{isSet}(X)$$

Contractible types, propositions and sets

- A is **contractible**

$$\mathsf{isContr}(A) \;:\equiv\; \sum_{x:A}\prod_{y:A} y = x$$

- A is a **proposition**

$$\mathsf{isProp}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isContr}(x = y)$$

- A is a **set**

$$\mathsf{isSet}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isProp}(x = y)$$

$$\mathsf{Prop} \;:\equiv\; \sum_{X:\mathsf{Type}} \mathsf{isProp}(X) \qquad \mathsf{Set} \;:\equiv\; \sum_{X:\mathsf{Type}} \mathsf{isSet}(X)$$

Univalence for Propositions and Sets

Immediate consequences of the univalence axiom:

Univalence for propositions

$P =_{\mathsf{Prop}} Q \;\simeq\; P \leftrightarrow Q$

Univalence for sets

$P =_{\mathsf{Set}} Q \;\simeq\; P \cong Q$

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. $M : \mathbf{Set}$
2. $\mu : M \times M \to M$
3. $e : M$
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) = \mu(a,\mu(b,c))$
5. $\lambda : \Pi_{(a:M)} \mu(e,a) = a$
6. $\rho : \Pi_{(a:M)} \mu(a,e) = a$

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. M : **Set**
2. $\mu : M \times M \to M$
3. $e : M$
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) = \mu(a,\mu(b,c))$
5. $\lambda : \Pi_{(a:M)} \mu(e,a) = a$
6. $\rho : \Pi_{(a:M)} \mu(a,e) = a$

Why M : **Set**?

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. $M : \mathbf{Set}$
2. $\mu : M \times M \to M$
3. $e : M$
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) = \mu(a,\mu(b,c))$
5. $\lambda : \Pi_{(a:M)} \mu(e,a) = a$
6. $\rho : \Pi_{(a:M)} \mu(a,e) = a$

Why $M : \mathbf{Set}$?

Abstractly, a monoid is a (dependent) pair $(\mathit{data}, \mathit{proof})$ where

- *data* is 1.–3.
- *proof* is 4.–6.

Structure Identity Principle

Univalence for monoids

$$M =_{\mathsf{Monoid}} N \quad \simeq \quad M \cong N$$

We also have univalence for other set-level strucuters (Coquand-Danielsson):

- groups, rings
- posets
- discrete fields
- sets with fixpoint operator

Structure Identity Principle

Univalence for monoids

$$M =_{\mathsf{Monoid}} N \quad \simeq \quad M \cong N$$

We also have univalence for other set-level strucuters (Coquand-Danielsson):

- groups, rings
- posets
- discrete fields
- sets with fixpoint operator

What about **categories**?

Univalence for categories

We only have univalence for **univalent** categories: ones where the canonical function $A = B \to A \cong B$ for objects $A, B : \mathscr{C}$ is an equivalence.

Here, the homsets are sets, and the type of objects will be groupoids.

Univalence for univalent categories

$$\mathscr{C} =_{\mathsf{UCat}} \mathscr{D} \;\simeq\; C \simeq D$$

We also have univalence for other higher strucuters (Ahrens-North-Shulman-Tsementzis):

- bicategories, tricategories, etc
- double categories
- dagger categories

Further resources

- HoTT Reading Group, 10:30-12 on Wednesdays
- HoTT Book
  - https://homotopytypetheory.org/book/
- Learn how to write proofs in a computer!
  - https://leanprover-community.github.io/learn.html
  - (Number Game)

Thank you!