Univalent Foundations and the equivalence principle

Benedikt Ahrens

Outline

Outline

Indiscernability of identicals

Leibniz's law

$$x = y \;\rightarrow\; \forall P\,(P(x) \leftrightarrow P(y))$$

- Reasoning **in logic** is invariant under equality
- **In mathematics**, reasoning should be invariant under weaker notion of sameness!

The equivalence principle

Equivalence principle

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

The equivalence principle

Equivalence principle

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

Notion of sameness 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 categories that is not invariant under the equivalence of categories

Non-examples: statements violating equivalence principle

We can easily **violate** this principle:

Exercise

Find a statement about categories that is not invariant under the equivalence of categories

A solution

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

Maybe this statement is simply silly!

A language for invariant properties

M. 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

M. 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 invariance principle for categories...

- Recall: the statement

  The category $\mathcal{C}$ 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 invariance principle for categories...

- Recall: the statement

  The category $\mathcal{C}$ 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 $\mathsf{source}(g)$ is **equal to** $\mathsf{target}(f)$, then $g \circ f$ exists."

How to break the invariance principle for categories...

- Recall: the statement

  The category $\mathcal{C}$ 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 $\mathsf{source}(g)$ is **equal to** $\mathsf{target}(f)$, then $g \circ f$ exists."

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

. . . and how to fix it.

Solution

Use a logic/language of **dependent types**, in which $s(g) = t(f)$ is encoded by what type of thing f and g are.

. . . and how to fix it.

Solution

Use a logic/language of **dependent types**, in which $s(g) = t(f)$ is encoded by what type of thing f and g are.

A category consists of

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

. . . and how to fix it.

Solution

Use a logic/language of **dependent types**, in which $s(g) = t(f)$ is encoded by what type of thing f and g are.

A category consists of

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

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

Invariance for properties

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 properties

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 properties

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?

Equivalence principle in the univalent foundations

In the univalent foundations

- an equivalence principle can be proved for a variety of structures
  - sets
  - groups, rings, ...
  - categories
- EP applies not only to properties, but also to constructions: any construction transports suitably along equivalence

Outline

Outline

Univalent foundations

Syntax/Language

- Language of **dependent types**, a.k.a., a type theory
- Logic and constructions are treated uniformly

Semantics/Models

- Natural semantics in **spaces** (simplicial sets)
- Other interpretations are possible

Syntax of UF

Fundamental: **judgment**

$$\text{context} \quad \vdash \quad \text{conclusion}$$

Contexts & judgments

Γ	sequence of variable declarations $x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(\vec{x}_i)$
$\Gamma \vdash A$	A is well–formed **type** in context Γ
$\Gamma \vdash a : A$	**term** a is well-formed and of type A
$\Gamma \vdash A \equiv B$	types A and B are **convertible**
$\Gamma \vdash a \equiv b : A$	a is convertible to b in type A

$$(x : \mathsf{Nat}), (f : \mathsf{Nat} \to \mathsf{Nat}) \quad \vdash \quad x^2 : \mathsf{Nat}$$

Rules and derivations

- A **rule** is an implication of judgments,

$$J_1 \wedge J_2 \wedge \ldots \wedge J_n \quad \Longrightarrow \quad J$$

  e.g.,

$$\Gamma \ \vdash \ a \equiv b : A \quad \Longrightarrow \quad \Gamma \ \vdash \ b \equiv a : A$$

  (often written as inference rule with horizontal bar)

- A **derivation** is a sequence of rules.

- Derivations are done **algorithmically**.
  E.g., the judgment $\Gamma \ \vdash \ a \equiv b : A$ is derived automatically (or fails to be derived)

- There is a different equality (**identity**) that can be proved or disproved, see later

- We sometimes omit the context when writing judgments.

About well-typedness

There is no way to derive a judgment of the form

$$(b : \mathsf{Bool}), (f : \mathsf{Nat} \to A) \quad \vdash \quad f(b) : A$$

We are hence not allowed to write $f(b)$ here.

Some differences to set-theoretic membership

- the judgment $a : A$ is **not** a statement that can be proved or disproved
- a valid term has exactly one type up to $\equiv$

Declaring types & terms

Any type (and corresponding term) construction is declared by giving 4 (groups of) rules:

Formation declaring the type

Introduction a way to construct terms of that type

Elimination what can one do with a term of that type

Computation what happens when one does Introduction followed by Elimination

The type of pairs $A \times B$

Formation $(\Gamma \vdash A) \wedge (\Gamma \vdash B) \implies \Gamma \vdash A \times B$

Introduction $(\Gamma \vdash a : A) \wedge (\Gamma \vdash b : B) \implies \Gamma \vdash \mathsf{pair}(a, b) : A \times B$

Elimination $\Gamma \vdash t : A \times B \implies \Gamma \vdash \mathsf{pr}_1(t) : A$
(and analogous for pr_2)

Computation $\mathsf{pr}_1(\mathsf{pair}(a, b)) \equiv a$ and $\mathsf{pr}_2(\mathsf{pair}(a, b)) \equiv b$

The type of functions $A \to B$

Formation $(\Gamma \vdash A) \wedge (\Gamma \vdash B) \implies \Gamma \vdash A \to B$

Introduction $\Gamma, (x : A) \vdash b(x) : B \implies \Gamma \vdash \lambda x.b(x) : A \to B$

Elimination $(\Gamma \vdash f : A \to B) \wedge (\Gamma \vdash a : A) \implies \Gamma \vdash f(a) : B$

Computation $(\lambda x.b)(a) \equiv b[x := a]$

- $\lambda x.b$ corresponds to $x \mapsto b$.
- **Substitution** $b[x := a]$ is built-in
- Example: $\emptyset \vdash \lambda x.x^2 : \mathsf{Nat} \to \mathsf{Nat}$

Type dependency

In particular: dependent type B over A

$$x : A \;\vdash\; B(x)$$

"family B of types indexed by A"

- a type can depend on several variables

The type of dependent functions $\prod_{(x:A)} B$

Formation $(\Gamma \vdash A) \wedge (\Gamma, x : A \vdash B(x)) \implies \Gamma \vdash \prod_{(x:A)} B(x)$

Introduction $\Gamma, (x : A) \vdash b : B \implies \Gamma \vdash \lambda x.b : \prod_{(x:A)} B$

Elimination $(\Gamma \vdash f : \prod_{(x:A)} B) \wedge (\Gamma \vdash a : A) \implies \Gamma \vdash f(a) : B[x := a]$

Computation $(\lambda x.b)(a) \equiv b[x := a]$

- The case $A \to B$ is a special case, where B does not depend on $x : A$

The type of dependent pairs $\sum_{(x:A)} B$

Formation $(\Gamma \vdash A) \wedge (\Gamma, x : A \vdash B(x)) \implies \Gamma \vdash \sum_{(x:A)} B(x)$

Introduction term $\mathsf{pair}(a, b)$ with $a : A$ and $b : B(a)$

Elimination given $t : \sum_{(x:A)} B$, there is $\mathsf{pr}_1(t) : A$ and

$$\mathsf{pr}_2(t) : B(x := \mathsf{pr}_1(t))$$

Computation $\mathsf{pr}_1(a, b) \equiv a$ and $\mathsf{pr}_2(a, b) \equiv b$

- The case $A \times B$ is a special case, where B does not depend on $x : A$

Logic in univalent foundations

Curry-Howard isomorphism:

- propositions are types
- proofs of P are terms of type P

Hence

- In principle, all types could be called propositions
- to prove a propositions P means to construct a term of type P
- In UF, only some types are called "propositions", cf later

The identity type

Formation $(\Gamma \vdash a : A) \wedge (\Gamma \vdash b : A) \implies \Gamma \vdash \mathsf{Id}_A(a, b)$

Introduction $\mathsf{refl}_a : \mathsf{Id}_A(a, a)$

Elimination $C(a, a, \mathsf{refl}_a) \implies C(a, b, p)$

Computation $\ldots$

We also write $a =_A b$ and $a = b$ for $\mathsf{Id}_A(a, b)$

More about identities

Can construct terms of type

- $a = b \to b = a$
- $a = b \times b = c \to a = c$
- $B(a) \times a = b \to B(b)$

Can **not** construct a term UIP of type

$$(x : A), (p : x = x) \;\vdash\; UIP : p =_{(x=x)} \mathsf{refl}_x$$

but can construct a term

$$(x : A), (p : \sum_{y:A} x = y) \;\vdash\; contr : p = \mathsf{pair}(x, \mathsf{refl}_x)$$

Types are ∞-groupoids

Garner, van den Berg

$$(A, \mathsf{Id}_A, \mathsf{Id}_{\mathsf{Id}_A}, \ldots)$$

forms ∞-groupoid, i.e., groupoid laws hold up to "higher" identities

- gives rise to model of type theory in simplicial sets (Voevodsky)
- this model motivates (and justifies) the univalence axiom, cf later

Contractible types, propositions and sets

Definitions:

- A is **contractible** if we can construct a term of type

$$\mathsf{isContr}(A) \stackrel{\text{def}}{=} \sum_{(x:A)} \prod_{(y:A)} y = x$$

- A is a **proposition** if $\prod_{(x\ y:A)} x = y$ is inhabited

- A is a **set** if, for any $x, y : A$, $x = y$ is a proposition

Equivalences

Definition

A map $f : A \to B$ is an **equivalence** if it has contractible fibers, i.e.,

$$\mathsf{isequiv}(f) \stackrel{\text{def}}{=} \prod_{b:B} \mathsf{isContr}\left(\sum_{a:A} f(a) = b\right)$$

The type of equivalences:

$$A \simeq B \stackrel{\text{def}}{=} \sum_{f:A\to B} \mathsf{isequiv}(f)$$

Characterizing some identity types

Can construct equivalences (i.e., terms of type)

- for $\mathsf{pair}(a, b) : A \times B$,

$$\Big(\mathsf{pair}(a, b) = \mathsf{pair}(a', b')\Big) \simeq \Big((a = a') \times (b = b')\Big)$$

- for $f, g : A \to B$

$$\Big(f = g\Big) \simeq \Big(\prod_{a:A} f(a) = g(a)\Big)$$

- ...

Outline

Universes

There is also a type $\mathcal{U}$ that contains all types, i.e., $A : \mathcal{U}$.

- Actually, hierarchy $(\mathcal{U}_i)_{i \in I}$ to avoid paradoxes.
- a dependent type

$$x : A \ \vdash \ B : \mathcal{U}$$

can be considered as a function

$$\lambda x.B : A \to \mathcal{U}$$

Question

What is

$$\mathsf{Id}_{\mathcal{U}}(A, B) \quad ?$$

Voevodsky's Univalence Axiom

Answer 1

$$\mathsf{univalence} : (A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

More controlled:

Answer 2

Define

$$\mathsf{idtoeqv} : \prod_{A,B:\mathcal{U}} (A = B) \to (A \simeq B)$$

$$\mathsf{refl}_A \mapsto \mathsf{id}$$

$$\mathsf{univalence} : \prod_{A,B:\mathcal{U}} \mathsf{isequiv}(\mathsf{idtoeqv}_{A,B})$$

Summary: Univalent Foundations

- A language of **dependent types**, a.k.a. a **type theory**
- With an interpretation in ∞-groupoids (Kan complexes)

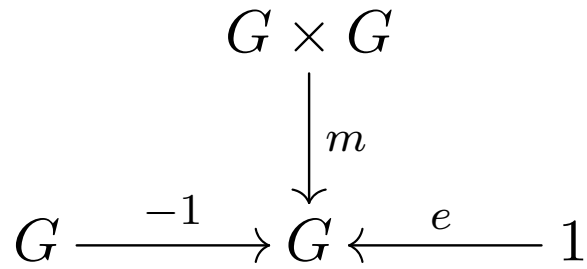
Type theory	Interpretation
A type	∞-groupoid A
$a : A$ (term a of type A)	object a of ∞-groupoid A
$f : A \to B$	∞-functor $A \to B$
$p : a =_A b$	path (1-morphism) from a to b in A
$\alpha : p =_{a=_A b} q$	homotopy from p to q in A

- Universe of **sets** given by **discrete groupoids**
- Logic and constructions are treated uniformly in UF

Outline

Groups in Univalent Foundations

A **group** in Univalent Foundations is



such that

- G is a **discrete** type, i.e. a "set"
- group axioms are satisfied

Lifting the equivalence principle from types to groups

A **group isomorphism** $G \to G'$ is

- a bijective function on the underlying types $G \to G'$
- compatible with the group structures on G and G'.

Lifting the equivalence principle from types to groups

A **group isomorphism** $G \to G'$ is

- a bijective function on the underlying types $G \to G'$
- compatible with the group structures on G and G'.

Theorem (EP on types lifts to EP on groups)

- *An isomorphism of groups lifts to an equivalence of all constructions on groups (in UF).*
- *In particular: any statement about groups is invariant under group isomorphism*

Lifting univalence from types to groups

The proof of this statement uses 2 ingredients:

1. $(G = G') \simeq (G \cong G')$
2. Leibniz's law (equality is substitutive)

$(G = G') \simeq (G \cong G')$ is given by the canonical map

$$\mathsf{refl}_G \mapsto \mathsf{id}_G$$

Lifting univalence to algebraic structures (Aczel, Coquand, Danielsson)

For many algebraic structures in univalent foundations, univalence lifts.

Lifting univalence to algebraic structures (Aczel, Coquand, Danielsson)

For many algebraic structures in univalent foundations, univalence lifts.

Examples include:

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

An equivalence principle for categories

Univalence for algebraic structures

For groups, rings, etc., univalence lifts.

An equivalence principle for categories

Univalence for algebraic structures

For groups, rings, etc., univalence lifts.

Going to equivalence of categories:

Univalence for categories (A., Kapulkin, Shulman)

For **univalent** categories, equivalence is identity via canonical map

$$(\mathcal{C} = \mathcal{D}) \quad \simeq \quad \mathsf{Equiv}(\mathcal{C}, \mathcal{D}) \quad .$$

An equivalence principle for categories

Univalence for algebraic structures

For groups, rings, etc., univalence lifts.

Going to equivalence of categories:

Univalence for categories (A., Kapulkin, Shulman)

For **univalent** categories, equivalence is identity via canonical map

$$(\mathcal{C} = \mathcal{D}) \quad \simeq \quad \mathsf{Equiv}(\mathcal{C}, \mathcal{D}) \quad .$$

That is, any construction on univalent categories in Univalent Foundations is invariant under **equivalence**.

An equivalence principle for categories

Univalence for algebraic structures

For groups, rings, etc., univalence lifts.

Going to equivalence of categories:

Univalence for categories (A., Kapulkin, Shulman)

For **univalent** categories, equivalence is identity via canonical map

$$(\mathcal{C} = \mathcal{D}) \quad \simeq \quad \mathsf{Equiv}(\mathcal{C}, \mathcal{D}) \quad .$$

That is, any construction on univalent categories in Univalent Foundations is invariant under **equivalence**.

- What is a category in UF?
- What is the **univalence** condition for categories?

Categories in univalent mathematics

A category is

- a type $O : \mathcal{U}$ of objects
- a dependent type $A : O \times O \to \mathcal{S}et$ of arrows
- $\mathsf{id} : \prod_{(a:O)} A(a, a)$
- $(\circ) : \prod_{(a,b,c:O)} A(a, b) \times A(b, c) \to A(a, c)$
- axioms postulating identities of arrows

Categories in univalent mathematics

A **univalent** category is

- a type $O : \mathcal{U}$ of objects
- a dependent type $A : O \times O \to \mathcal{S}et$ of arrows
- $\mathsf{id} : \prod_{(a:O)} A(a, a)$
- $(\circ) : \prod_{(a,b,c:O)} A(a, b) \times A(b, c) \to A(a, c)$
- axioms postulating identities of arrows
- such that the natural map

$$\mathsf{idtoiso} : \prod_{a,b:O} (a = b) \to \mathsf{iso}(a, b)$$

is an equivalence for any a, b

Some remarks on univalent categories

In simplicial set model

- categories correspond to truncated Segal spaces (Rezk, A model for the homotopy theory of homotopy theory)
- univalence corresponds to completeness

Connection with Freyd, Blanc

Instead of avoiding identity of objects as in [F,B], in a univalent category, identity means (can be replaced with) isomorphism.

Examples of univalent categories

- $\mathcal{S}et$ (discrete types)
- Groups, rings, ... (Structure Identity Principle)
- Functor category $[\mathcal{C}, \mathcal{D}]$, if $\mathcal{D}$ is univalent
- Full subcategories of univalent categories

More examples of univalent categories

- A preorder is univalent iff it is antisymmetric
- If X is of h-level 3, i.e., 1-truncated, then there is a univalent category with X as objects and $\hom(x, y) := (x = y)$
- If $\mathcal{C}$ is univalent, then the category of cones of shape $F : \mathcal{J} \to \mathcal{C}$ is
  - $\rightsquigarrow$ limits (limiting cones) in a univalent category are unique **up to identity**

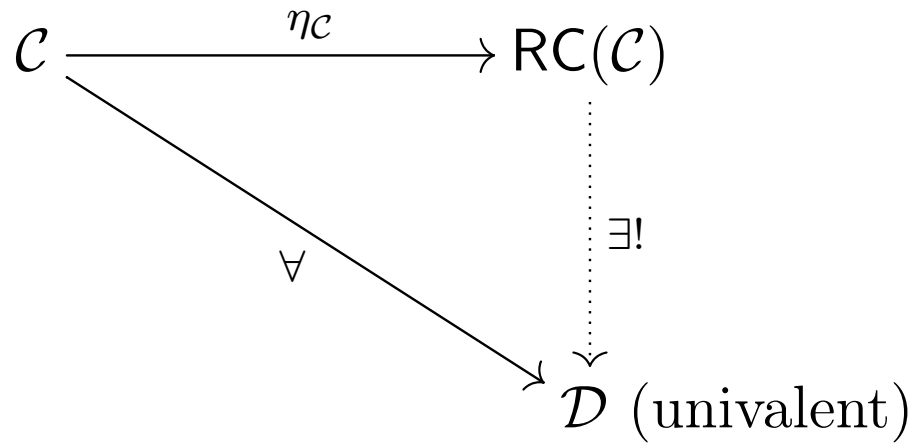
Non-univalent categories

- Any "chaotic" category $\mathcal{C}$ with $\mathcal{C}(x, y) := 1$, for $\mathcal{C}_0$ not a prop

- Any chaotic category $\mathcal{C}$ with an object $c : \mathcal{C}_0$ is **equivalent** to the terminal category $\mathbf{1}$
  - $\rightsquigarrow$ a category can be equivalent to a univalent one without being univalent itself

Rezk completion

To any category $\mathcal{C}$, associate a univalent one, its "Rezk completion",



Intuitively, obtain $\mathsf{RC}(\mathcal{C})_0$ by adding to $\mathcal{C}_0$ as many identities as needed

Construction of the Rezk completion

- $\mathsf{RC}(\mathcal{C})$ is the full image subcategory of the Yoneda embedding $\mathcal{C} \to [\mathcal{C}^{\mathrm{op}}, \mathcal{S}et]$
- $\eta_{\mathcal{C}} : \mathcal{C} \to \mathsf{RC}(\mathcal{C})$ is fully faithful and essentially surjective
- precomposition with a ff. and es. functor is ff. and es.
- a ff. and es. functor is an equivalence if source category is univalent
- the object map of an equivalence of univalent categories is an equivalence of types

Some references

- Rezk, *A model for the homotopy theory of homotopy theory*
- Freyd, *Properties invariant within equivalence types of categories*
- Blanc, *Equivalence naturelle et formules logiques en théorie des catégories*
- Coquand, Danielsson, *Isomorphism is equality*
- Kapulkin, Lumsdaine, Voevodsky, *The Simplicial Model of Univalent Foundations*
- Gambino, van den Berg, *Types are weak ω-groupoids*
- Ahrens, Kapulkin, Shulman, *Univalent categories and the Rezk completion* `http://arxiv.org/abs/1303.0584`

Item			
Syntax	Set interpretation	Logic	Simpl. set interpretation
A	set A	proposition A	Kan complex A
$(A, \mathsf{Id}_A, \mathsf{Id}_{\mathsf{Id}_A}, \ldots)$			
$a : A$	$a \in A$	a is a proof of A	$a \in A_0$
$A \times B$	cartesian product	$A \wedge B$	binary product
$A \to B$	set of functions	$A \Rightarrow B$	space of maps
$A + B$	disjoint union	$A \vee B$	binary coproduct
$x : A \ \vdash \ B(x)$	family B of sets indexed by A	predicate B on A	fibration $B \to A$ with fibers $B(x)$
$\sum_{(x:A)} B(x)$	disjoint union	$\exists x, B(x)$	total space of fibration $B \to A$
$\prod_{(x:A)} B(x)$	dependent function	$\forall x, B(x)$	space of sections of fib. $B \to A$
$\mathsf{Id}_A(a, b)$	equality $a = b$		$\hom_A(a, b)$ or $A_1(a, b)$