

# Univalent foundations and the equivalence principle

Benedikt Ahrens      Paige Randall North

In this paper, we explore the ‘equivalence principle’ (EP): roughly, statements about mathematical objects should be invariant under an appropriate notion of equivalence for the kinds of objects under consideration. In set theoretic foundations, EP may not always hold: for instance, the statement ‘ $1 \in \mathbb{N}$ ’ is not invariant under isomorphism of sets. In univalent foundations, on the other hand, EP has been proven for many mathematical structures. We first give an overview of earlier attempts at designing foundations that satisfy EP. We then describe how univalent foundations validates EP.

## 1 The equivalence principle

What should it mean for two objects  $x$  and  $y$  to be equal? One proposal by Leibniz [11], known as the “identity of indiscernibles”, states that if  $x$  and  $y$  have the same properties, then they must be equal:

$$\forall \text{ properties } P, (P(x) \leftrightarrow P(y)) \rightarrow x = y.$$

For this proposal to be reasonable, then the converse, the “indiscernibility of identicals,” should hold incontrovertibly. That is, if  $x$  and  $y$  are equal, then they must have the same properties:

$$x = y \rightarrow \forall \text{ properties } P, (P(x) \leftrightarrow P(y)). \quad (1)$$

Indeed, one would be hard-pressed to find a mathematician who disagreed with this principle. However, in classical mathematics based on set theory, this principle is of limited usefulness: too few objects are equal. A group theorist, for example, would have little interest in a principle which required them to suppose that two groups are equal.

Instead, mathematicians are often interested in weaker notions of sameness and those properties that are invariant under such notions. A group theorist, for example, would have more interest in an analogous principle that described the properties of any pair of isomorphic groups  $G$  and  $H$ :

$$G \cong H \rightarrow \forall \text{ group theoretic properties } P, (P(G) \leftrightarrow P(H)).$$

Similarly, category theorists would be more interested in a principle that described the properties of any pair of equivalent categories  $A$  and  $B$ :

$$A \simeq B \rightarrow \forall \text{ category theoretic properties } P, (P(A) \leftrightarrow P(B)).$$

To generalize: mathematicians working in some domain  $\mathcal{D}$  often utilize a stronger variant of the principle given in line (1) above, called the **equivalence principle**: for all objects  $x$  and  $y$  of domain  $\mathcal{D}$ :

$$x \sim_{\mathcal{D}} y \rightarrow \forall \mathcal{D}\text{-properties } P, (P(x) \leftrightarrow P(y)) \quad , \quad (2)$$

where  $\sim_{\mathcal{D}}$  denotes a suitable notion of sameness for the domain  $\mathcal{D}$ .

We might consider a still stronger variant of the equivalence principle. A group theorist, for example, might not only want *properties of* groups to be invariant under isomorphism, but they might also want *structures on* groups to be invariant under isomorphism. For example, if the equivalence principle (2) holds in the domain of group theory and if two groups  $G$  and  $H$  are isomorphic, then the statements “ $G$  has a representation on  $V$ ” and “ $H$  has a representation on  $V$ ” are equivalent (for some fixed vector space  $V$ ). However, it is actually the case that the isomorphism  $G \cong H$  induces a bijection between the set of representations of  $G$  on  $V$  and the set of representations of  $H$  on  $V$ , which we regard as structures on  $G$  and  $H$  respectively. Such a variant of the equivalence principle has become known as the *Structure Identity Principle* (see [6],[15, Section 9.8], [4]).

Our goal in this paper is to describe how one can find the right notion  $\sim_{\mathcal{D}}$  of sameness and the right class of ‘ $\mathcal{D}$ -properties and  $\mathcal{D}$ -structures’ for some specific domains  $\mathcal{D}$ .

This right notion of sameness is not uniformly defined across different mathematical objects. However, we usually use the one already present in mathematical practice since we aim for the equivalence principle to capture mathematical practice. As a rule of thumb, it is usually considered to be

- *equality* when the objects naturally form a set—numbers, functions, etc.
- *isomorphism* when the objects naturally form a category—sets, groups, etc.
- *equivalence* when the objects naturally form a bicategory—e.g., categories.

The hard part will be in determining the right class of  $\mathcal{D}$ -properties and  $\mathcal{D}$ -structures for some specific domain  $\mathcal{D}$ . In usual mathematical practice, we can state properties which break the equivalence principle; that is, we can state properties of mathematical objects that are not invariant under sameness. We will seek to exclude such properties from our class of  $\mathcal{D}$ -properties and  $\mathcal{D}$ -structures.

*Exercise 1.* Denote by  $2\mathbb{N}$  the set of even natural numbers. Find a property of sets that is not invariant under the isomorphism  $\mathbb{N} \cong 2\mathbb{N}$  given by multiplying and dividing by 2, respectively.

**Answer:** One such statement is given in the abstract.

*Exercise 2.* Find a property of categories that is true for one, but not for the other of these two, equivalent, categories.



**Answer:** The statement “The category  $\mathcal{C}$  has exactly one object.” is such a statement.

Thus, to assert an equivalence principle for sets or categories, we need to exclude these properties from our collection of ‘set theoretic properties’ and ‘category theoretic properties’. M. Makkai [13] says

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.

Put differently, establishing an equivalence principle means establishing a *syntactic criterion* for properties and structures that are invariant under sameness.

## 2 History

Look again at Example 2. There, we violated the equivalence principle for categories by referring to equality of objects. This might lead one to conjecture (correctly) that categorical properties which obey the equivalence principle cannot mention equality of objects.

However, the traditional definition of category mentions equality of objects. It usually includes the following axiom: for any two morphisms  $f$  and  $g$  such that the codomain of  $f$  equals the domain of  $g$ , there is a morphism  $gf$  such that domain of  $gf$  equals the domain of  $f$  and the codomain of  $gf$  equals the codomain of  $g$ .

To avoid mentioning equality of objects, one can express the composability of morphisms of that category via different means, specifically by having not one collection of morphisms but many *hom-sets*: one for each pair of objects. This idea, for instance explained in [12, Section I.8] usually requires asking the hom-sets to be disjoint. This last requirement is automatic if we work instead in a typed language, where types are automatically disjoint.

A category is then given by

- a type  $O$  of objects,
- for each  $x, y : O$ , a type  $A(x, y)$  of arrows from  $x$  to  $y$ ,
- for each  $x, y, z : O$  and  $f : A(x, y)$ ,  $g : A(y, z)$ , a composite arrow  $g \circ f : A(x, z)$ , and
- for each  $x : O$ , an identity arrow  $\text{id}_x : A(x, x)$

such that

- for each  $w, x, y, z : O$  and  $f : A(w, x)$ ,  $g : A(x, y)$ ,  $h : A(y, z)$ , there is an equality  $h \circ (g \circ f) = (h \circ g) \circ f$  in  $A(w, z)$ ,
- for each  $x, y : O$  and  $f : A(x, y)$ , there is an equality  $f \circ \text{id}_x = f$  in  $A(x, y)$ , and
- for each  $x, y : O$  and  $f : A(x, y)$ , there is an equality  $\text{id}_y \circ f = f$  in  $A(x, y)$ .

Note that when stating axioms, the only equality that is mentioned is the equality within a hom-set of the form  $A(x, y)$ , that is, between arrows of the same type.

By adding quantifiers, ranging over one type at a time, to this typed language, we obtain a language for stating properties of, and constructions on, categories. It turns out that the statements of that language are invariant under equivalence of categories:

**Theorem 3** (*Théorème de préservation par équivalence* [5]). *A property of categories (expressed in 2-typed first order logic) is invariant under equivalence if and only if it can be expressed in the typed language sketched above, and without referring to equality of objects.*

We do not give here the precise form of the typed language, but refer instead to Blanc’s article for details. Note that Freyd [9] states a similar result to Blanc’s above, in terms of “diagrammatic properties”.

Makkai [13] develops notions of *signature* and *theory*, to specify mathematical structures. A theory is a pair  $(L, \Sigma)$  consisting of a signature  $L$  (specifying the shape of the structure) and a set  $\Sigma$  of “axioms” over  $L$  (specifying the axioms of the structure). A theory determines a notion of “model”—which is an  $L$ -structure satisfying the properties specified by  $\Sigma$ —and of “equivalence” of such models, called  $L$ -equivalence.

His *Invariance Theorem* gives a result similar to Theorem 3 for models of a theory: given an interpretation  $T = (L, \Sigma) \rightarrow S$  of such a theory in a first-order logic theory, an  $S$ -sentence  $\phi$  is invariant under  $L$ -equivalence if and only if it is expressible in First Order Logic with Dependent Sorts (FOLDS) over  $L$ .

In the following sections, we will see that similar results can be shown in univalent foundations. Specifically, not only properties but also constructions will be “invariant” under equivalence, and invariance of properties will be recovered as a special case via the propositions-as-some-types correspondence.

### 3 Univalent foundations and transport of structures along equivalences

Starting in the 1970’s, Per Martin-Löf designed several versions of dependent type theory, which are now called Martin-Löf Type Theories [14]. These were intended to be foundations of mathematics that, unlike set theory, have an inherent notion of computation built in. For decades, Martin-Löf type theories have formed the basis of computer proof assistants such as Coq and Agda.

One of the most mysterious features of this kind of type theory is its *equality* type  $a =_X b$  of any two inhabitants  $a$  and  $b$  of a type  $X$ —see [3, Section 4.3]. Inhabitants of such an equality type behave, in many ways, like a proof of equality; in particular, they can be composed and inverted, corresponding to the transitivity and symmetry of equality. In one important respect, however, they behave differently: as explained in [3, Sections 4.3 and 5.1], one can *not* show that any two inhabitants  $e, f$  of an equality type  $a =_X b$  are equal—with their equality now being given by the iterated equality type  $e =_{a=_X b} f$ .

The lack of uniqueness of those terms has given rise to a new way of thinking about them and interpreting them into the world of mathematical objects. Instead of interpreting them as (set-theoretic) equalities between  $a$  and  $b$  in the set interpreting  $X$ , one can interpret them as *paths* from  $a$  to  $b$  in a space interpreting  $X$ .

This intuition is made formal in Voevodsky’s *simplicial set model* [10] which satisfies an additional interesting property: given two types  $X$  and  $Y$ , the interpretation of their equality type  $X = Y$  is equivalent to the interpretation of their type of equivalences  $X \simeq Y$  (see [3, Section 5.4]). This observation motivated Voevodsky to add this property as an axiom to Martin-Löf type theory which he called the *Univalence Axiom*. The addition of the Univalence Axiom turns Martin-Löf type theory into univalent foundations.

Obtaining an equivalence principle was one of the main motivations for Voevodsky in designing his univalent foundations:

[...] My homotopy lambda calculus is an attempt to create a system which is very good at dealing with equivalences. In particular it is supposed to have the property that given any type expression  $F(T)$  depending on a term subexpression  $t$  of type  $T$  and an equivalence  $t \rightarrow t'$  (a term of the type  $Eq(T; t, t')$ ) there is a mechanical way to create a new expression  $F'$  now depending on  $t'$  and an equivalence between  $F(T)$  and  $F'(T')$  (note that to get  $F'$  one can not just substitute  $t'$  for  $t$  in  $F$  – the resulting expression will most likely be syntactically incorrect). [Email to Daniel R. Grayson, Sept 2006]

In the following sections, we describe how Voevodsky’s goal is realized in univalent foundations.

### 3.1 Indiscernibility of identicals in type theory

In Martin-Löf type theory (perhaps without the univalence axiom), identicals—that is, elements  $x, y : T$  with an equality  $e : x =_T y$  between them—are easily seen to be indiscernible. That is, for every type  $T$  and  $x, y : T$ , we can find a function

$$x =_T y \rightarrow \forall \text{ properties } P, (P(x) \leftrightarrow P(y)) \quad . \quad (3)$$

To better formulate this in the language of dependent type theory, (i) we will define this function for all  $x, y$  at once, (ii) we will understand ‘properties  $P$ ’ to be functions

$P : T \rightarrow \mathbf{Type}$ , and (iii) we replace the logical equivalence  $P(x) \leftrightarrow P(y)$  with the type-theoretic equivalence

$$(P(x) \simeq P(y)) := \left( \sum_{f:P(x) \rightarrow P(y)} \text{isEquiv}(f) \right)$$

(where  $\text{isEquiv}$  is defined in Section 5.4 of Altenkirch’s introduction [3]).

Our goal is hence to define a function

$$\text{transport} : \prod_{(x,y:T)} \left( x =_T y \rightarrow \prod_{P:T \rightarrow \mathbf{Type}} (P(x) \simeq P(y)) \right). \quad (4)$$

To this end, recall that in order to define a map out of an equality type, it suffices to define its image on  $\text{refl}_x : (x =_T x)$  for each  $x : T$ . Therefore, it suffices to show that there is a term

$$\text{transport}(\text{refl}) : \prod_{(x:T)} \left( \prod_{P:T \rightarrow \mathbf{Type}} (P(x) \simeq P(x)) \right).$$

But then, for each  $x : T$  and  $P : T \rightarrow \mathbf{Type}$ , we can set this to be the equivalence  $1_{P(x)} : P(x) \simeq P(x)$  whose underlying function  $P(x) \rightarrow P(x)$  is the identity function:

$$\text{transport}(\text{refl})(x)(P) \equiv 1_{P(x)}.$$

The function  $\text{transport}$  shows that any ‘property’, or dependent type,  $P : T \rightarrow \mathbf{Type}$  is invariant under equalities in  $T$ . In particular, given an equality  $x =_T y$ , we obtain functions  $P(x) \rightarrow P(y)$  and  $P(y) \rightarrow P(x)$  which allow us to transport terms of  $P(x)$  or  $P(y)$  back and forth along this equality.

### 3.2 From equality to equivalence

We have just seen that in Martin-Löf type theory, identicals are indiscernible. Now we investigate how to expand this to get a full-blown equivalence principle from this fact. In short, we will see that in many circumstances, the equality type  $x =_T y$  is itself equivalent to some structured equivalence appropriate for the type  $T$ . Then composing this equivalence with the  $\text{transport}$  function, we will obtain the equivalence principle (2).

To be precise, fix a type  $T$ . Given any notion  $\sim_T$  of equivalence (or at least a reflexive relation) in a type  $T$ , we immediately obtain a function

$$\text{idtoequiv} : \prod_{x,y:T} ((x = y) \rightarrow (x \sim_T y)) \quad (5)$$

by setting  $\text{idtoequiv}(x,x)(\text{refl})$  to the reflexive term on  $x$  in  $x \sim x$  (since to define a function out of an equality type, it is enough to define it just at every occurrence of  $\text{refl}$ ).

Now we hope that for notions of equivalence  $\sim_T$  already of interest to us, this function is actually an equivalence for all terms  $x, y : T$ , or more precisely, that the following type is inhabited.

$$\prod_{x, y : T} \text{isEquiv}(\text{idtoequiv}_{x, y})$$

If this type is indeed inhabited, then for each  $x, y : T$  we can take  $\pi_1(\text{isEquiv}(\text{idtoequiv}(x, y)))$ , the backwards function  $(x \sim_T y) \rightarrow (x = y)$ , and compose it with `transport` to obtain a function

$$\prod_{(x, y : T)} \left( x \sim_T y \rightarrow \prod_{P : T \rightarrow \text{Type}} (P(x) \simeq P(y)) \right)$$

which is our equivalence principle.

Thus, in the next sections, we just aim to show that for certain types  $T$  and notions of equivalence  $\sim_T$ , the function `idtoequiv` is indeed an equivalence.

### 3.3 The univalence principle

“Equality is equivalence for types” is the slogan made precise by Voevodsky’s univalence principle. More precisely, the univalence principle asserts part of an equivalence principle for types: it states that the canonical map

$$\text{idtoequiv} : \prod_{A, B : \text{Type}} ((A = B) \rightarrow (A \simeq B)) \tag{6}$$

from equalities of types to equivalences of types is itself, for any types  $A$  and  $B$ , an equivalence. Then, composing `idtoequiv` with `transport` as in the last section, we obtain an equivalence principle for types.

$$\prod_{(A, B : \text{Type})} \left( A \simeq B \rightarrow \prod_{P : \text{Type} \rightarrow \text{Type}} (P(A) \simeq P(B)) \right)$$

The univalence principle is not provable in pure Martin-Löf type theory [14], but needs to be postulated as an axiom—hence it is sometimes also called the “univalence axiom”. In extensions of Martin-Löf type theory, as in the recently developed cubical type theory [7], the univalence principle can be derived.

Building upon the equivalence principle for types—whether it is given as an axiom or as a theorem—one can derive equivalence principles for other kinds of structures. Establishing that `idtoequiv` is an equivalence for other types and notions of equivalence is the subject of the next sections.

## 4 The equivalence principle for set-level structures

Now we turn our attention away from the type of all types and towards types of more specific mathematical objects. It turns out that for types of simple objects like propositions,

sets, and monoids, the univalence axiom is enough to show the equivalence principle for these types' usual notion of equivalence. More precisely, in the presence of the univalence axiom, the function `idtoequiv` discussed in the last section is itself an equivalence. For an exploration and formalization of these ideas, see [8].

## 4.1 Propositions

We call *propositions* those types that have at most one inhabitant. We think of propositions as either being *true* (when they are inhabited) or *false* (when they are not inhabited). What should an equivalence of two propositions  $P, Q$  be? Experience might indicate that such an equivalence should just be two functions

$$f : P \hookrightarrow Q : g$$

so that  $P$  is inhabited if and only if  $Q$  is. In fact, this notion of equivalence is the right one in the sense that it will validate the equivalence principle.

To be precise, we define

$$\begin{aligned} \text{isProp} &: \text{Type} \rightarrow \text{Type} \\ \text{isProp } A &:\equiv \prod_{(x,y:A)} x = y \end{aligned}$$

whose inhabitants can be thought of as proofs that a type  $A$  is a proposition. A proposition is then a pair  $(A, p)$  of a type  $A$  and a proof  $p : \text{isProp } A$ , that is,

$$\text{Prop} :\equiv \sum_{A:\text{Type}} \text{isProp } A .$$

Now we have, for  $P \equiv (A, p)$  and  $Q \equiv (B, q)$ ,

$$\begin{aligned} P = Q &\simeq (A, p) = (B, q) \\ &\simeq \sum_{(e:A=B)} (\text{transport}^{\text{isProp}}(e, p)) = q \end{aligned} \tag{7}$$

$$\simeq \sum_{(e:A=B)} 1 \tag{8}$$

$$\simeq A = B \tag{9}$$

$$\simeq (A \rightarrow B) \times (B \rightarrow A) \tag{10}$$

Equivalence (7) above uses the fact that an equality between pairs is the same as pairs of equalities, where the second equality is “heterogeneous”, i. e., requires a transport along the first equality to make it well-typed. Equivalence (8) uses the fact that being a proposition is itself a proposition, so that equality types between proofs of a proposition are equivalent to the unit type. Equivalence (9) is given by the univalence principle,



and equivalence (10) uses that  $A$  and  $B$  are propositions; a pair of maps back and forth between types that are propositions automatically forms an equivalence of types.

Altogether, this means that  $P$  and  $Q$  are equal exactly if their underlying types  $A$  and  $B$  are logically equivalent—the expected notion of equivalence for propositions.

## 4.2 Sets

We call *sets* those types whose equality types are propositions,

$$\text{Set} := \sum_{A:\text{Type}} \prod_{x,y:A} \text{isProp}(x = y).$$

Then a set in the type theory is a collection of terms, the equality types among which are either empty or contractible.

Given two sets  $X, Y : \text{Set}$ , where  $X = (A, p)$  and  $Y = (B, q)$ , the equivalences of types

$$(X = Y) \simeq (A \simeq B) \tag{11}$$

$$\simeq \sum_{f:A \cong B} \text{isCoherent}(f) \tag{12}$$

$$\simeq (A \cong B) \tag{13}$$

can be constructed. Here,  $A \cong B := \sum_{(f:A \rightarrow B)} \text{isIso} f$  is the type of *isomorphisms of types* between  $A$  and  $B$ , and  $\text{isCoherent}(f)$  states an equality of equalities in  $A$ . When  $A$  is a set, the type  $\text{isCoherent}(f)$  is contractible (see the discussion in [3, Section 5.4]), and hence we obtain equivalence (13).

## 4.3 Monoids

The equivalence principle can be shown for many algebraic structures commonly encountered in mathematics, such as groups and rings. Before presenting a general result to that extent in Section 4.4, in this section, we study in detail the case of monoids (which was formalized in [8]). This particular case exemplifies many of the concepts and results used in general.

A *monoid* is a tuple  $(M, \mu, e, \alpha, \lambda, \rho)$  where

1.  $M : \text{Set}$
2.  $\mu : M \times M \rightarrow M$  (multiplication)
3.  $e : 1 \rightarrow M$  (neutral element)
4.  $\alpha : \prod_{(a,b,c:M)} \mu(\mu(a, b), c) = \mu(a, \mu(b, c))$  (associativity)
5.  $\lambda : \prod_{(a:M)} \mu(e, a) = a$  (left neutrality)
6.  $\rho : \prod_{(a:M)} \mu(a, e) = a$  (right neutrality)

Given two monoids  $\mathbf{M} \equiv (M, \mu, e, \alpha, \lambda, \rho)$  and  $\mathbf{M}' \equiv (M', \mu', e', \alpha', \lambda', \rho')$ , a *monoid isomorphism* is a bijection  $f : M \cong M'$  between the underlying sets that preserves multiplication and neutral element. We can derive an equivalence between the equality type and the isomorphism type between any two monoids as follows:

$$\mathbf{M} = \mathbf{M}' \simeq (M, \mu, e) = (M', \mu', e') \tag{14}$$

$$\begin{aligned} &\simeq \sum_{p:M=M'} (\text{transport}^{Y \mapsto (Y \times Y \rightarrow Y)}(p, \mu) = \mu') \\ &\quad \times (\text{transport}^{Y \mapsto (1 \rightarrow Y)}(p, e) = e') \\ &\simeq \sum_{f:M \cong M'} (f \circ m \circ (f^{-1} \times f^{-1}) = m') \\ &\quad \times (f \circ e = e') \\ &\simeq \mathbf{M} \cong \mathbf{M}' \end{aligned} \tag{15}$$

Here, the equivalence of types (14) uses the fact that the axioms of a monoid (the types of  $\alpha$ ,  $\lambda$ , and  $\rho$ ) are propositions (compare also to (8) above). The equivalence (15) uses the univalence principle for types in the first component, replacing an equality of sets by a bijection. This translates to replacing “transport along the equality” by “conjugating by the bijection” in the second component.

The equivalence of types constructed above, from left to right, is pointwise equal to the canonical map

$$\prod_{\mathbf{M}, \mathbf{M}'} \mathbf{M} = \mathbf{M}' \rightarrow \mathbf{M} \cong \mathbf{M}' \tag{16}$$

defined by equality elimination, which shows that the latter is an equivalence of types. In other words, we have just proved the equivalence principle (5) for the equivalence  $\mathbf{M} \cong \mathbf{M}'$ .

#### 4.4 Univalent categories

We have seen in the preceding sections that the types of propositions, sets, and monoids all have a certain nice property – they validate the equivalence principle. However, it is natural to consider such objects as each belonging to a category. In this section, we discuss those categories whose objects validate the equivalence principle.

In Section 2, we saw that in order to avoid mentioning equality of objects, we can define a *category*  $A$  to consist of

1. a type  $A_0$  of *objects*;
2. for each  $a, b : A_0$ , a set  $A(a, b)$  of *arrows* or *morphisms*;
3. for each  $a : A_0$ , a morphism  $1_a : A(a, a)$ ;

4. for each  $a, b, c : A_0$ , a function of type

$$A(a, b) \times A(b, c) \rightarrow A(a, c)$$

denoted by  $(f, g) \mapsto f \cdot g$ ;

5. for each  $a, b : A_0$  and  $f : A(a, b)$ , we have  $\ell_f : f = 1_a \cdot f$  and  $r_f : f = f \cdot 1_b$ ;
6. for each  $a, b, c, d : A_0$  and  $f : A(a, b)$ ,  $g : A(b, c)$ ,  $h : A(c, d)$ , we have  $\alpha_{f,g,h} : f \cdot (g \cdot h) = (f \cdot g) \cdot h$ .

The reason for asking the types of arrows to be *sets* rather than arbitrary types is so that these categories behave as classical categories (and not any kind of higher category) and, in particular, so that the axioms—which state equalities between arrows—are propositions, meaning that we do not need to state higher coherence axioms. There is *prima facie* no condition of that kind on the type of objects of the category. However, it will turn out that the objects of a *univalent category* form a groupoid (meaning that all of its equality types form sets).

A morphism  $f : A(a, b)$  of the category  $A$  is an *isomorphism* if there is a morphism  $g : A(b, a)$  that is left and right inverse to  $f$ , that is

$$\text{isIso}f := \sum_{g:A(b,a)} (f \cdot g = 1_a) \times (g \cdot f = 1_b) .$$

We call  $\text{Iso}(a, b) := \sum_{f:A(a,b)} \text{isIso}f$  the type of isomorphisms from  $a$  to  $b$ , and for any  $a : A_0$  we have  $1_a : \text{Iso}(a, a)$ . We can define a function

$$\text{idtoiso} : \prod_{x,y:A_0} (x = y) \rightarrow \text{Iso}(x, y) \tag{17}$$

by setting  $\text{idtoiso}_{x,x}(\text{refl}_x)$  to  $1_x$  for every  $x : A_0$  just as we did to define  $\text{idtoequiv}$  in Section 3.2.

Now we call the category  $A$  *univalent* if  $\text{idtoiso}_{x,y}$  is an equivalence of types for every  $x, y : A_0$ . To see why the adjective *univalent* is used, compare the function in Display (17) above to the one in Display (6) underlying the univalence principle. The univalence principle asserts that equality and equivalence of types are the same; here, we assert that equality and isomorphism of objects of a category are the same.

In asserting that a category  $A$  is univalent, we assert that the equality types  $a = b$  among its objects are equivalent to the sets  $\text{Iso}(a, b)$  of isomorphisms among its objects. Since the property of “being a set” itself obeys the equivalence principle for types the equality types  $a = b$  are themselves sets. When a type’s equality types are sets, we call the type a *groupoid*.

A categorical equivalence between univalent categories  $A$  and  $B$  gives rise to an isomorphism between them—indeed, the type  $A \simeq B$  of adjoint equivalences is equivalent to the type  $A \cong B$  of isomorphisms of categories.

With a set-theoretic reading of the univalence condition in mind, one could think that only skeletal categories are univalent. However, one should keep in mind that in type

theory, the equality type  $x = y$  between two objects of a category can—and often does—have more than one element. Consequently, in type theory, a category being univalent usually signifies that its type of objects has many equalities. This difference is witnessed by the many examples of univalent categories given below, most of which are not skeletal.

With these definitions in place, the composite equivalence of types shown in Displays (11) - (13) can be restated as “the category of sets is univalent”. Similarly, the result of Section 4.3 can be restated as “the category of monoids is univalent”.

Many categories that arise naturally are univalent, in particular,

- the category of sets;
- the categories of groups, rings, etc.;
- the functor category  $[A, B]$  if the target category  $B$  is;
- a preorder, seen as a category, exactly if it is anti-symmetric.

To extend our list of univalent categories to other algebraic structures beyond monoids, we could simply redo constructions similar to those for monoids, for groups, rings, and other structures of interest. However, in doing so, we would observe that we are doing the same reasoning over and over again. For instance, looking back at monoids, we used that the category of sets is univalent to show that the category of monoids is univalent, in step (15). This is due to the fact that “monoids are sets with additional structure”, and monoid isomorphisms are isomorphisms of sets preserving this structure. Similarly, “groups are monoids with additional structure”, and we would expect to reuse the equivalence of Display (16) when building an equivalence between the equality types of groups on the one hand, and of group isomorphisms on the other hand. *Displayed categories* as presented in [2] are a convenient tool for such modular reasoning about categories built step-by-step from simpler ones. In particular, Proposition 43 and Theorem 44 of [2] allow one to show that a category built from a simpler one using the framework of displayed categories is univalent, provided the simpler one is univalent and the “extra data” making the difference between the two categories satisfies some condition. That result validates the *Structure Identity Principle* [15, Theorem 9.8.2].

## 5 The equivalence principle for (higher) categorical structures

We saw in the previous sections that for types of simple structures like propositions, sets, and monoids, the equivalence principle comes along with the univalence axiom. Now we see that for more complicated structures, like categories, the equivalence principle only holds for certain well-behaved categories.

The most common notion of equivalence between two categories  $A$  and  $B$  is unsurprisingly called an equivalence  $A \simeq B$ . It consists of two functors  $F : A \rightarrow B$  and  $G : B \rightarrow A$  and natural isomorphisms  $1_B \cong FG$  and  $1_A \cong GF$  (see [12]). An equivalence of categories “transports” categorical structures, such as limits, between categories, and is hence considered the right notion of sameness for categories in most contexts. Can we show that

the equality type between two categories,  $A = B$ , is the same as the type of categorical equivalences  $A \simeq B$ ? The answer is that while this is not the case for arbitrary categories, it is the case when  $A$  and  $B$  are univalent.

For any two categories  $A$  and  $B$ , the univalence axiom implies that the function from equalities to isomorphisms (a stricter notion of sameness of categories) given by equality elimination is an equivalence [1, Lemma 6.16]:

$$(A = B) \xrightarrow{\simeq} (A \cong B) \tag{18}$$

Furthermore, if  $A$  and  $B$  are univalent categories, then the type of isomorphisms between them is equivalent to that of categorical equivalences [1, Lemma 6.15]:

$$(A \cong B) \xrightarrow{\simeq} (A \simeq B) \tag{19}$$

Composing these two equivalences yields the desired equivalence of types between equalities and categorical equivalences.

The example of categories shows that, in order to obtain the equivalence principle for mathematical structures that naturally form bicategory, one needs to impose a “univalence” condition on those structures. Defining such a univalence condition for general structures is the subject of active research.

**Acknowledgments** We are very grateful to Deniz Sarikaya and Deborah Kant for their editorial work and their encouragement, and to an anonymous referee for providing valuable feedback. Furthermore, we would like to thank all the organizers of the FOMUS workshop—Balthasar Grabmayr, Deborah Kant, Lukas Kühne, Deniz Sarikaya, and Mira Viehstädt—for giving us the opportunity to discuss and compare different foundations of mathematics.

This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-16-1-0212 and FA9550-17-1-0363.

## References

- [1] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science*, 25:1010–1039, 2015.
- [2] Benedikt Ahrens and Peter LeFanu Lumsdaine. Displayed categories (*conference version*). In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16. Leibniz-Zentrum für Informatik, 2017.
- [3] Thorsten Altenkirch. Naïve type theory. In Stefania Centrone, Deborah Kant, and Deniz Sarikaya, editors, *Reflections on the Foundations of Mathematics: Univalent Foundations, Set Theory and General Thoughts*, pages 101–136. Springer International Publishing, Cham, 2019.

- [4] Steve Awodey. Structuralism, Invariance, and Univalence. *Philosophia Mathematica*, 22(1):1–11, 10 2013.
- [5] Georges Blanc. Équivalence naturelle et formules logiques en théorie des catégories. *Arch. Math. Logik Grundlag.*, 19(3-4):131–137, 1978/79.
- [6] Samuel Buss, Ulrich Kohlenbach, and Michael Rathjen. Oberwolfach Reports – Mathematical Logic: Proof Theory, Constructive Mathematics. pages 2963–3002. <https://doi.org/10.4171/OWR/2011/52>.
- [7] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [8] Thierry Coquand and Nils Anders Danielsson. Isomorphism is equality. *Indagationes Mathematicae*, 24(4):1105 – 1120, 2013. In memory of N.G. (Dick) de Bruijn (1918–2012).
- [9] Peter Freyd. Properties invariant within equivalence types of categories. In *Algebra, topology, and category theory (a collection of papers in honor of Samuel Eilenberg)*, pages 55–61. Academic Press, New York, 1976.
- [10] Chris Kapulkin and Peter LeFanu Lumsdaine. The Simplicial Model of Univalent Foundations (after Voevodsky). *J. Eur. Math. Soc.* arXiv:1211.2851.
- [11] Gottfried Wilhelm Leibniz. *Philosophical Papers and Letters*, volume 2 of *Synthese Historical Library*. Springer-Verlag, 1989.
- [12] Saunders Mac Lane. *Categories for the working mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1998.
- [13] Michael Makkai. Towards a categorical foundation of mathematics. In *Logic Colloquium '95 (Haifa)*, volume 11 of *Lecture Notes Logic*, pages 153–190. Springer, Berlin, 1998.
- [14] Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.
- [15] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. <http://homotopytypetheory.org/book>, Institute for Advanced Study, 2013.