The Integers as a Higher Inductive Type

Thorsten Altenkirch*
School for Computer Science
University of Nottingham
Nottingham, United Kingdom
txa@cs.nott.ac.uk

Luis Scoccola†
Department of Mathematics
University of Western Ontario
London, Ontario, Canada
lscoccol@uwo.ca

Abstract

We consider the problem of defining the integers in Homotopy Type Theory (HoTT). We can define the type of integers as signed natural numbers (i.e., using a coproduct), but its induction principle is very inconvenient to work with, since it leads to an explosion of cases. An alternative is to use set-quotients, but here we need to use set-truncation to avoid non-trivial higher equalities. This results in a recursion principle that only allows us to define function into sets (types satisfying UIP). In this paper we consider higher inductive types using either a small universe or bi-invertible maps. These types represent integers without explicit set-truncation that are equivalent to the usual coproduct representation. This is an interesting example since it shows how some coherence problems can be handled in HoTT. We discuss some open questions triggered by this work. The proofs have been formally verified using cubical Agda.

CCS Concepts: • **Theory of computation → Categorical semantics**; **Type theory**.

Keywords: higher inductive type, coherence problem, truncation, initiality

ACM Reference Format:
Thorsten Altenkirch and Luis Scoccola. 2020. The Integers as a Higher Inductive Type. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3373718.3394760

*Supported by EPSRC grant EP/M016994/1 and by USAF, Airforce office for scientific research, award FA9550-16-1-0029.
†Partially supported by Mitacs Globalink Research Award IT14154

LICS '20, July 8–11, 2020, Saarbrücken, Germany

ACM ISBN 978-1-4503-7104-9/20/07…$15.00
https://doi.org/10.1145/3373718.3394760

1 Introduction

How to define the integers in HoTT? This can sound like a trivial question. The first answer is as signed natural numbers:

Definition 1.1. Let $\mathbb{Z}_w$ be the inductive type generated by the following constructors:

- $0 : \mathbb{Z}_w$
- $\mathsf{strpos} : \mathbb{N} \to \mathbb{Z}_w$
- $\mathsf{strneg} : \mathbb{N} \to \mathbb{Z}_w$

However, this type is very inconvenient in practice because it creates a lot of unnecessary case distinctions. Nuo [15] tried to prove distributivity of multiplication over addition, which resulted in a lot of cases. It is like working with normal forms only, when working with λ-terms.

Nuo shows that it is much better to work with a quotient type, representing integers as differences of natural numbers. That is, we define $\mathbb{Z}_q = \mathbb{N} \times \mathbb{N}/\sim$ where $(x^+, x^-) \sim (y^+, y^-)$ is defined as $x^+ + y^- = y^+ + x^-$ [1]. However, this is not the end of the story. Here we use set-quotients, which can be implemented as a higher inductive type with a set-truncation constructor [18, Section 6.10]. However, the set-truncation constructor implies that using its recursion principle we can only define functions into sets, which seems to be an unreasonable limitation when working in HoTT. For example, in the proof that the loop space of the circle is isomorphic to the integers [17], we must map from the integers to the loop space of the circle, when we do not yet know that this will end up being a set.

We would like to have a definition of the integers which is convenient to work with (i.e., does not reduce them to normal forms) but which is not forced to be set-truncated by a set-truncation constructor. Paolo Capriotti suggested the following definition:

Definition 1.2. Let $\mathbb{Z}_h$ be the higher inductive type with the following constructors:

- $0 : \mathbb{Z}_h$;
- $\mathsf{succ} : \mathbb{Z}_h \to \mathbb{Z}_h$;
- $\mathsf{pred} : \mathbb{Z}_h \to \mathbb{Z}_h$;
- $\mathsf{sec} : (z : \mathbb{Z}_h) \to \mathsf{pred}(\mathsf{succ}(z)) = z$;
- $\mathsf{ret} : (z : \mathbb{Z}_h) \to \mathsf{succ}(\mathsf{pred}(z)) = z$;
- $\mathsf{coh} : (z : \mathbb{Z}_h) \to \mathsf{ap}_{\mathsf{succ}}(\mathsf{sec}(z)) = \mathsf{ret}(\mathsf{succ}(z))$.

[1] This is actually the definition in [18].

We add succ and pred as constructors, but then we postulate that they are inverse to each other using sec and ret. At this point we could add a set-truncation but then we would suffer from the same shortcoming as the definition using a set-quotient. However, we can add just one coherence condition coh which should look familiar to anybody who has read the HoTT book: indeed the constructors pred, sec, ret, and coh exactly say that succ is a half-adjoint equivalence [18, Section 4.2]. More precisely, sec postulates that succ is a section, ret postulates that succ is a retraction, and coh represents the triangle identity in the definition of half-adjoint equivalence.

The question that now remains is the following. Is $\mathbb{Z}_h$ a correct definition of the integers, in particular is it a set with decidable equality? The strategy to prove this is to define a normalisation function into the signed integers, $\mathbb{Z}_w$, and show that this normalisation function, together with the obvious embedding of $\mathbb{Z}_w$ into $\mathbb{Z}_h$, forms an equivalence. It turns out that this is actually quite hard to prove, due to the presence of higher equalities, and nobody has so far been able to formally verify this.

In this paper, we follow the same idea but use a *simpler* definition of equivalence, namely bi-invertible maps [18, Section 4.3]:

Definition 1.3. Let $\mathbb{Z}_b$ be the higher inductive type with the following constructors:

- $0 : \mathbb{Z}_b$;
- $\mathsf{succ} : \mathbb{Z}_b \to \mathbb{Z}_b$;
- $\mathsf{pred}_1 : \mathbb{Z}_b \to \mathbb{Z}_b$;
- $\mathsf{pred}_2 : \mathbb{Z}_b \to \mathbb{Z}_b$;
- $\mathsf{sec} : (z : \mathbb{Z}_b) \to \mathsf{pred}_1(\mathsf{succ}(z)) = z$;
- $\mathsf{ret} : (z : \mathbb{Z}_b) \to \mathsf{succ}(\mathsf{pred}_2(z)) = z$;

In this case we postulate that succ has a left inverse, given by pred_1 and sec, and a right inverse, given by pred_2 and ret. The reason why $\mathbb{Z}_b$ is simpler than $\mathbb{Z}_h$ is because it only has 0- and 1-dimensional constructors. The higher coherence coh is not needed in this case for the same reason that a 2-dimensional constructor is not needed in the definition of bi-invertible map: having two, a priory, unrelated inverses makes the type of witnesses that a certain map is bi-invertible a proposition ([18, Theorem 4.3.2]).

For this definition we can give a complete proof that $\mathbb{Z}_b$ is equivalent to $\mathbb{Z}_w$, which has been formalized in cubical Agda. We remark that this has previously been verified by Evan Cavallo [7] in RedTT [1]. However, our approach to prove the equivalence is more general. Our main result is Theorem 2.4, which says that only the components witnessing the preservation of 0 and succ are relevant when comparing morphisms out of $\mathbb{Z}_b$.

Another presentation of the integers follows from directly implementing the idea that the integers can be specified as the initial type with an inhabitant and an equivalence:

- $0 : \mathbb{Z}_U$;
- $s : \mathbb{Z}_U = \mathbb{Z}_U$.

The problem is that this is not a standard definition of a higher inductive type because we state an equality of the type itself. However, this can be fixed by using a small universe:

Definition 1.4. Define $U : \mathcal{U}$ and $\mathsf{El} : U \to \mathcal{U}$ inductively with the constructors:

- $z : U$;
- $q : z = z$;
- $0 : \mathsf{El}(z)$

Now, let $\mathbb{Z}_U :\equiv \mathsf{El}(z)$.

While we can show that this is a set without using set-truncation, its recursion principle isn't directly amenable to recursive definitions of functions because even succ is not a constructor. On the other hand the fact that the integers are the loop space of the circle is a rather easy consequence of this definition.

The definition of the integers is also closely related to the free group, indeed as suggested in [13] we can define the free group over a type A by simply parametrizing all the constructors but 0:

Definition 1.5. Given $A : \mathcal{U}$, define $\mathbf{F}(A)$ inductively with the constructors:

- $0 : \mathbf{F}(A)$;
- $\mathsf{succ} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$;
- $\mathsf{pred}_1 : A \to \mathbf{F}(A) \to \mathbf{F}(A)$;
- $\mathsf{pred}_2 : A \to \mathbf{F}(A) \to \mathbf{F}(A)$;
- $\mathsf{sec} : (a : A) \to (z : \mathbf{F}(A)) \to \mathsf{pred}_1(a, \mathsf{succ}(a, z)) = z$;
- $\mathsf{ret} : (a : A) \to (z : \mathbf{F}(A)) \to \mathsf{succ}(a, \mathsf{pred}_2(a, z)) = z$;

The integers arise as the special case $\mathbb{Z} = \mathbf{F}(\mathbf{1})$. However, the normal forms get a bit more complicated because we must allow alternating sequences of succ and pred but only for different $a : A$. This means that a normalisation function is only definable for sets $a : A$ with a decidable equality. The general problem of whether $\mathbf{F}(A)$ is a set, if A is, is still open — in [13] it is shown to be the case, if we 1-truncate the HIT.

The problem of defining the integers with convenient constructors, and adding only the right coherences to make it a set, can be seen as a simple instance of a more general class of coherence problems in HoTT. Another example that we have in mind is the intrinsic definition of the syntax of type theory as the initial category with families as developed in [3]. If we carry out this definition in HoTT, we need to set-truncate the syntax, but this stops us from interpreting the syntax in the standard model formed by sets. We hope that also in this case we can add the correct coherence laws and show that they are sufficient to deduce that the initial algebra is a set.

1.1 Contributions

We show that the definitions of the signed integers, $\mathbb{Z}_w$, the definition of the integers as a higher inductive type using

bi-invertible maps, $\mathbb{Z}_b$, and the definition using a higher inductive-inductive type with a mini universe, $\mathbb{Z}_U$, are all equivalent (Theorem 4.7).

For $\mathbb{Z}_b$ we establish some useful principles such as a recursion principle (Proposition 2.1) which only uses one predecessor, and an induction principle which says that to prove a predicate (i.e., a family of propositions), you only need to prove closure under 0, succ, and pred_1 (Proposition 2.2). This is sufficient to verify all algebraic properties of the integers, e.g., that the integers form a commutative ring. We have formalized [5] the constructions using cubical Agda [2].

When formalizing the constructions involving $\mathbb{Z}_b$ we developed the theory of bi-invertible maps in cubical Agda, which wasn't available. In particular, we prove that bi-invertible maps are equivalent to contractible-fibers maps [18, Section 4.4], and the principle of equivalence induction for bi-invertible maps.

1.2 Related work

The claim that $\mathbb{Z}_h$ is a set can be found in [4] but the proof was flawed: it relies on the assumption that we can ignore propositional parts of an algebra for a certain signature when constructing algebra morphisms, which is not the case in general (Example 6.1). Cavallo [7] verified that $\mathbb{Z}_b \simeq \mathbb{Z}_w$ in RedTT. Higher inductive representations of the integers are discussed in [6] and it is shown there that $\mathbb{Z}_h$ without the last constructor is not a set. [14] also discuss [4] and note that it is a corollary of their higher Seifert-van Kampen theorem — however, they derive it from initiality not from the induction principle.

1.3 Background

We use Homotopy Type Theory as presented in the book [18]. We adopt the following notational conventions.

If two terms a and b are definitionally equal, we write $a \equiv b$, and we reserve $a = b$ to denote the type of propositional equalities between a and b.

Given a type $A : \mathcal{U}$ and a type family $P : A \to \mathcal{U}$, we write the corresponding Π-type as $(a : A) \to P(a)$, and the corresponding Σ-type as $(a : A) \times P(a)$.

Given a type $A : \mathcal{U}$, a type family $P : A \to \mathcal{U}$, an equality $e : a = b$ in A, and $p : P(a)$, we denote the coercion of p along e by $e_*(p) : P(b)$. This is defined by induction on e.

A type is contractible if it has exactly one inhabitant. That is, given a type $A : \mathcal{U}$, we define $\mathsf{isContr}(A) :\equiv (a_0 : A) \times ((a : A) \to a = a_0)$. A type is a proposition if any two inhabitants are equal. That is, given a type $A : \mathcal{U}$, we define $\mathsf{isProp}(A) :\equiv (a, b : A) \to a = b$, [18, Definition 3.3.1]. A type is a set if it satisfies UIP. That is, given a type $A : \mathcal{U}$, we define $\mathsf{isSet}(A) :\equiv (a, b : A) \to (p, q : a = b) \to p = q$, [18, Definition 3.1.1].

An equivalence between types A and B is a map $f : A \to B$ together with a proof that $(b : B) \to \mathsf{isContr}((a : A) \times f(a) = b)$. The type of equivalences between A and B is denoted by $A \simeq B$.

The general syntax of Higher Inductive Inductive Types (HIITs) is specified in [11], where also the types of the eliminators are derived. In the informal exposition, and in the formalisation, we use the cubical approach to path algebra introduced in [16].

For the formalisation we use cubical Agda [2] which is based on the cubical type theory of [8]. The development of HIITs in Agda is based on [9].

2 Representing $\mathbb{Z}$ using bi-invertible maps

The type $\mathbb{N}$ of natural numbers is usually defined as the inductive type generated by an inhabitant $0 : \mathbb{N}$ and an endomap $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$. In this section, we define the integers $\mathbb{Z}_b$ in a similar way. The idea is to give constructors that guarantee that we have $0 : \mathbb{Z}_b$, $\mathsf{succ} : \mathbb{Z}_b \to \mathbb{Z}_b$, and that succ is an equivalence using bi-invertible maps, see Definition 1.3. To make it easy to work with this definition, we prove three theorems that let us: map out of $\mathbb{Z}_b$ (Proposition 2.1), prove properties about $\mathbb{Z}_b$ (Proposition 2.2), and recognise when two maps out of $\mathbb{Z}_b$ are equal (Theorem 2.4).

The result about mapping out of $\mathbb{Z}_b$ is very simple, and follows immediately from the recursion principle of $\mathbb{Z}_b$.

Proposition 2.1 (`rec`$\mathbb{Z}$`simp`). *Given a type T with an inhabitant $t : T$ and two maps $f : T \to T$, $g : T \to T$, such that g is a left and right inverse of f, we get a map $r : \mathbb{Z}_b \to T$ such that $r(0) \equiv t$ and $r(\mathsf{succ}(z)) \equiv f(r(z))$, definitionally.*

The next result is only slightly more involved.

Proposition 2.2 (`ind`$\mathbb{Z}$`simp`). *Given a type family $P : \mathbb{Z}_b \to \mathcal{U}$ such that $(z : \mathbb{Z}_b) \to \mathsf{isProp}(P(z))$, if we have $P(0)$, $(z : \mathbb{Z}_b) \to P(z) \to P(\mathsf{succ}(z))$, and $(z : \mathbb{Z}_b) \to P(z) \to P(\mathsf{pred}_1(z))$, then it follows that $(z : \mathbb{Z}_b) \to P(z)$.*

Proof. We use the induction principle of $\mathbb{Z}_b$. The main idea is that we do not have to check any coherences, since we are proving a proposition. Concretely, this means that we only have to provide inhabitants for the following types: $P(0)$, $(z : \mathbb{Z}_b) \to P(z) \to P(\mathsf{succ}(z))$, $(z : \mathbb{Z}_b) \to P(z) \to P(\mathsf{pred}_1(z))$, and $(z : \mathbb{Z}_b) \to P(z) \to P(\mathsf{pred}_2(z))$. For the first three we just use the assumptions. For the fourth one, we make use of the fact that, for every $z : \mathbb{Z}_b$, there is an equality $\mathsf{pred}_1(z) = \mathsf{pred}_2(z)$. This is because $\mathsf{pred}_2(z) = \mathsf{pred}_1(\mathsf{succ}(\mathsf{pred}_2(z))) = \mathsf{pred}_1(z)$ using sec and then ret. □

The result that allows us to compare maps out of $\mathbb{Z}_b$ is considerably more complicated to prove. In order to explain its proof, we need to talk about bi-invertible maps.

Definition 2.3. A map between types $f : A \to B$ is a bi-invertible map if there exist $g, h : B \to A$, and homotopies $s : g \circ f = \mathsf{id}_A$ and $r : f \circ h = \mathsf{id}_B$.

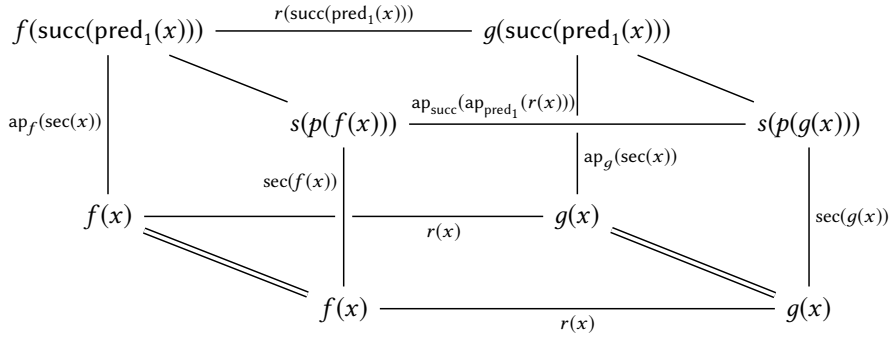

Figure 1. Cube needed for lemma 2.6

The type of bi-invertible structures on such a map f is denoted by $\mathsf{isBiInv}(f)$. The type $(f : A \to B) \times \mathsf{isBiInv}(f)$ is denoted by $A \simeq_b B$.

Whenever we have $f : A \simeq_b B$, we will abuse notation, and write $f : A \to B$ for the underlying function of the bi-invertible map f.

Notice that the constructors succ, pred_1, pred_2, sec, and ret form a bi-invertible map. Suppose given a type T with an inhabitant $t : T$ and a bi-invertible map $s : T \simeq_b T$. The recursion principle of $\mathbb{Z}_b$ gives us $\mathrm{rec}_{\mathbb{Z}_b}(T, t, s) : \mathbb{Z}_b \to T$. Now, assume given another map $f : \mathbb{Z}_b \to T$. What do we have to check to be able to conclude that $f = \mathrm{rec}_{\mathbb{Z}_b}(T, t, s)$?

The following theorem gives a simple answer to the question and is the main focus of this section.

Theorem 2.4 ($\mathsf{uniqueness}\mathbb{Z}$)**.** *Given a type T, an inhabitant $t : T$, a bi-invertible map $s : T \simeq_b T$, and a map $f : \mathbb{Z}_b \to T$, if $f(0) = t$ and $s \circ f = f \circ \mathrm{succ}$ then $f = \mathrm{rec}_{\mathbb{Z}_b}(T, t, s)$.*

In order to prove Theorem 2.4 we must study the preservation of bi-invertible maps, which we introduce next.

Fix types $A, B, A', B' : \mathcal{U}$, bi-invertible maps $e : A \simeq_b B$ and $e' : A' \simeq_b B'$, and maps $\alpha : A \to A'$ and $\beta : B \to B'$:

$$\begin{array}{ccc} A & \xrightarrow{e} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A' & \xrightarrow{e'} & B'. \end{array}$$

We now define what it means for α and β to respect e and e'. By a slight abuse of notation, let the bi-invertible maps e and e' be given by (e, g, h, s, r) and (e', g', h', s', r').

Definition 2.5. We define the type $\mathsf{prBiInv}(e, e', \alpha, \beta)$ as the iterated Σ-type with the following fields:

- (preservation of e) $p_e : e' \circ \alpha = \beta \circ e$;
- (preservation of g) $p_g : g' \circ \beta = \alpha \circ g$;
- (preservation of h) $p_h : h' \circ \beta = \alpha \circ h$;
- (preservation of s) $p_s : (a : A) \to s'(\alpha(a)) = \mathrm{ap}_{g'}(p_e a) \cdot p_g(e(a)) \cdot \mathrm{ap}_\alpha(s(a))$;
- (preservation of r) $p_r : (b : B) \to r'(\beta(b)) = \mathrm{ap}_{e'}(p_h b) \cdot p_e(h(b)) \cdot \mathrm{ap}_\beta(r(a))$.

The next proposition follows from the initiality of $\mathbb{Z}_b$, although it is a bit involved to prove formally using the constructors and the induction principle.

Proposition 2.6 ($\mathsf{uniqueness}$)**.** *Suppose given a type T with an inhabitant $t : T$, a bi-invertible map $s : T \simeq_b T$, and a map $f : \mathbb{Z}_b \to T$. If $f(0) = t$ and $\mathsf{prBiInv}(\mathrm{succ}, s, f, f)$, then $f = \mathrm{rec}_{\mathbb{Z}_b}(T, t, s)$.*

Proof. We write g for $\mathrm{rec}_{\mathbb{Z}_b}(T, t, s)$. By function extensionality, it is enough to construct a term $r : \Pi_{x:\mathbb{Z}_b} f(x) = g(x)$. We do this using the induction principle. The case for 0 follows directly from the assumption $f(0) = t$, and $r(\mathrm{succ}(x)) = \mathrm{ap}_s(g(x))$ and the corresponding equalities for pred_1 and pred_2 follow directly from the assumption that f respect the bi-invertible maps succ and s.

It remains to check the cases of sec and ret. Since these are symmetric, we only describe the case of sec. In this case, we have to provide a filler for the following square of equalities:

$$\begin{array}{ccc} f(\mathrm{succ}(\mathrm{pred}_1(x))) & \xrightarrow{r(\mathrm{succ}(\mathrm{pred}_1(x)))} & g(\mathrm{succ}(\mathrm{pred}_1(x))) \\ {\scriptstyle\mathrm{ap}_f(\mathrm{sec}(x))}\downarrow & & \downarrow{\scriptstyle\mathrm{ap}_g(\mathrm{sec}(x))} \\ f(x) & \xrightarrow[r(x)]{} & g(x). \end{array}$$

This filler can be obtained by filling the cube in figure 1, as follows. All the sides apart from the square in question can be filled using the fact that f preserves the bi-invertible maps, and simple path algebra, so we can conclude the proof using the Kan filling property of cubes: any open box can be filled. □

Given a type $A : \mathcal{U}$, let $\mathrm{id}_A : A \to A$ be the identity function. We have $\mathrm{idb}_A : \mathsf{isBiInv}(\mathrm{id}_A)$, so we can define a map $\mathrm{toBiInv} : A = B \to A \simeq_b B$ by path induction, sending $\mathrm{refl} : A = A$ to idb_A. By [18, Corollary 4.3.3] and the univalence axiom, the map toBiInv is an equivalence. Let $\mathrm{toEq} : A \simeq_b B \to A = B$ be its inverse.

From this we can derive the principle of (based) equivalence induction, which we now state.

Lemma 2.7 (BiInduction)**.** *Fix a type $A : \mathcal{U}$ and a type family $P : (B : \mathcal{U}) \to A \simeq_b B \to \mathcal{U}$. If we have $P_0 : P(A, \mathrm{id}_A, \mathrm{idb}_A)$, then we have $(B : \mathcal{U}) \to (e : A \simeq_b B) \to P(B, e)$.*

Proof. This is proven by path induction, after translating bi-invertible maps to equalities, using toEq and toBiInv. □

Using equivalence induction, and singleton elimination, one can finally prove that a map between types together with bi-invertible maps that respects the maps, automatically respects the bi-invertible structure.

Lemma 2.8. *The type* $\mathsf{prBiInv}(e, e', \alpha, \beta)$ *is equivalent to the type $e' \circ \alpha = \beta \circ e$.*

Proof. We use equivalence induction (Lemma 2.7) for e and e' and then observe that the type

$$\mathsf{prBiInv}((\mathsf{id}, \mathsf{idb}), (\mathsf{id}, \mathsf{idb}), \alpha, \beta)$$

is equivalent to the type of equalities $\alpha = \beta$. □

Proof of Theorem 2.4. The theorem is a corollary of Proposition 2.6 and Lemma 2.8. □

One should notice that Lemma 2.8 can be proven directly, avoiding the usage of the univalence axiom (which was used to prove that toBiInv is an equivalence). The reason why we don't do this, is because the path algebra involved in proving Lemma 2.8 directly is non-trivial.

3 $\mathbb{Z}$ is a set

In this section we relate $\mathbb{Z}_b$ with the usual definition of the integers as signed natural numbers, which we call $\mathbb{Z}_w$. We show that $\mathbb{Z}_b \simeq \mathbb{Z}_w$, and since we already know that $\mathbb{Z}_w$ is a set, we deduce that $\mathbb{Z}_b$ is a set too.

Definition 3.1. Let $\mathbb{Z}_w$ be the inductive type with the following constructors:

- $0 : \mathbb{Z}_w$
- $\mathsf{strpos} : \mathbb{N} \to \mathbb{Z}_w$
- $\mathsf{strneg} : \mathbb{N} \to \mathbb{Z}_w$

Theorem 3.2 (ℤisℤ)**.** *We have an equivalence $\mathbb{Z}_b \simeq \mathbb{Z}_w$.*

Proof. On the one hand, one can define $\mathsf{succ}_w : \mathbb{Z}_w \to \mathbb{Z}_w$ by induction, by mapping:

- $0 \mapsto \mathsf{strpos}(0)$;
- $\mathsf{strpos}(n) \mapsto \mathsf{strpos}(\mathsf{succ}(n))$;
- $\mathsf{strneg}(0) \mapsto 0$;
- $\mathsf{strneg}(\mathsf{succ}(n)) \mapsto \mathsf{strneg}(n)$.

Similarly one defines pred_w. The fact that pred_w provides a left and right inverse for succ_w is straightforward. So, by Proposition 2.1 we get a map $\mathsf{nf} : \mathbb{Z}_b \to \mathbb{Z}_w$. On the other hand, it is easy to construct a map $i : \mathbb{Z}_w \to \mathbb{Z}_b$ by induction.

Induction on $\mathbb{Z}_w$ shows that $\mathsf{nf} \circ i = \mathrm{id}_{\mathbb{Z}_w}$. The hard part is to show that $i \circ \mathsf{nf} = \mathrm{id}_{\mathbb{Z}_b}$. This is where Theorem 2.4 comes in handy. Theorem 2.4 implies that it is enough to check that $(i \circ \mathsf{nf})(0) = 0$ and that $\mathsf{succ} \circ (i \circ \mathsf{nf}) = (i \circ \mathsf{nf}) \circ \mathsf{succ}$, and this follows directly by construction. □

4 Representing $\mathbb{Z}$ using a universe

In this section we give another definition of the integers, denoted by $\mathbb{Z}_U$, which allows one to easily prove that they are the initial type together with an inhabitant and an equality from the type to itself.

To make sense of initiality, we first define the type of $\mathbb{Z}$-algebras and of $\mathbb{Z}$-algebra morphisms.

Definition 4.1. A $\mathbb{Z}$-algebra is a type $T : \mathcal{U}$ together with an inhabitant $t : T$, and an equality $e : T = T$. We denote such a $\mathbb{Z}$-algebra as (T, t, e), or T if the rest of the structure can be inferred from the context.

Definition 4.2. A morphism of $\mathbb{Z}$-algebras from (T, t, e) to (T', t', e') is given by a map $f : T \to T'$, together with an equality $f(t) = t'$, and a proof that $e_*(f) = e'_*(f)$. We denote the type of morphisms of $\mathbb{Z}$-algebras between T and T' by $T \to_{\mathbb{Z}} T'$.

We are interested in initial $\mathbb{Z}$-algebras.

Definition 4.3. A initial $\mathbb{Z}$-algebra is a $\mathbb{Z}$-algebra (T, t, e) such that for any other $\mathbb{Z}$-algebra (T', t', e') the type $T \to_{\mathbb{Z}} T'$ is contractible.

See Definition 1.4 for the definition of the initial $\mathbb{Z}$-algebra using a mini universe[2]. Then define an interpretation function $\mathsf{El} : U \to \mathcal{U}$, as the higher inductive family with only one constructor $0 : \mathsf{El}(z)$. Define the type $\mathbb{Z}_U :\equiv \mathsf{El}(z)$. The type $\mathbb{Z}_U$ has the structure of a $\mathbb{Z}$-algebra, since we have $0 : \mathbb{Z}_U$ and $s :\equiv \mathrm{ap}_{\mathsf{El}}(q) : \mathbb{Z}_U = \mathbb{Z}_U$. The following result follows by a routine application of the induction principle of $\mathbb{Z}_U$.

Theorem 4.4 (ℤisInitial)**.** *The $\mathbb{Z}$-algebra $\mathbb{Z}_U$ is initial.* □

In particular, we have.

Proposition 4.5. *Given a type T with an inhabitant $t : T$ and an equality $e : T = T$, we get a morphism of $\mathbb{Z}$-algebras $\mathbb{Z}_U \to T$.* □

Again, comparing maps out of $\mathbb{Z}_U$ is easy, thanks to the following theorem.

Theorem 4.6. *Given a type T, an inhabitant $t : T$, an equality $e : T = T$, and a map $f : \mathbb{Z}_U \to T$, if $f(0) = t$ and $e_* \circ f = f \circ s_*$ then $f = \mathrm{rec}_{\mathbb{Z}_U}(T, t, e)$.* □

[2] This is inspired by Zongpu Xie's proposal how to represent HIITs in Agda[10].

Analogously to the case of $\mathbb{Z}_b$, this is proven by combining the initiality of $\mathbb{Z}_U$ with the fact that to preserve an equality in the universe $e : T = T$, it is enough to commute with its corresponding coercion function $e_* : T \to T$.

Following the argument given in Section 3, one deduces the following.

Theorem 4.7. *There is an equivalence* $\mathbb{Z}_U \simeq \mathbb{Z}_w$. □

We omit the proof since it is basically the same as the construction presented in [17] when proving that the integers are the loop space of the circle.

Indeed, the mini universe U is nothing but the higher inductive type presentation of the circle S^1 of [18, Section 6.1], so that $(z = z) \equiv \Omega S^1$. Moreover, the type family El is equivalent to the path space fibration of the circle, in the following sense.

Theorem 4.8 (ElisPath)**.** *For every* $u : U$ *we have* $\mathsf{ed}(u) : \mathsf{El}(u) \simeq (z = u)$.

Proof. We construct a map $\mathsf{ed}(u) : \mathsf{El}(u) \to (z = u)$ using induction on U and mapping $0 : \mathsf{El}(z)$ to refl_z, To construct a map going the other way, we use path induction and map refl_z to 0. It is then straightforward to see that these maps give an equivalence as in the statement. □

As a corollary, we obtain the well-known equivalence between the loop space of the circle and the integers.

Corollary 4.9. *We have an equivalence* $\Omega S^1 \simeq \mathbb{Z}_w$. □

This suggests that alternatively one could view the representation of the integers as a universe as an inductive-inductive presentation of the circle equipped with a family that has a point in the fiber over the base point.

5 Formalization in cubical Agda

We formally checked the results of this paper [5] using cubical Agda [2]. There are two differences between the informal presentation in the paper and the formalisation. The first one is that the presentation in the paper is done using book-HoTT [18], whereas the formalisation is done using a cubical type theory. In this case, this difference is not important, since it is easy to translate the formalized arguments to book-HoTT.

The real difference is in the definition of higher inductive types. In the paper we define higher inductive types as initial algebras for a certain signature (Section 1.3). In the formalisation, we use higher inductive types as implemented in cubical Agda, which are based on [9]. Although it is natural to assume that the Agda higher inductive type should be initial in the sense of Section 1.3, proving this fact is actually one of the main difficulties in the formalisation (Proposition 2.6).

In proving the results of Section 2, we developed the theory of bi-invertible maps in cubical Agda, which wasn't available. We prove that the type of bi-invertible maps between A and B is equivalent to the type of equivalences between A and B, and the principle of bi-invertible induction.

6 Open questions

Preservation of properties

The key result in the above discussion is Lemma 2.8, which can be reformulated as follows. Let $T, T' : \mathcal{U}$, $s : T \to T$, $s' : T' \to T'$, $\phi : \mathsf{isBiInv}(s)$, and $\phi' : \mathsf{isBiInv}(s')$. We can define the following two types of morphisms between T and T':

$$\mathsf{Map}_{\mathsf{end}}(T, T') :\equiv (f : T \to T') \times (s' \circ f = f \circ s)$$
$$\mathsf{Map}_{\mathsf{biInv}}(T, T') :\equiv (f : T \to T') \times \mathsf{prBiInv}(s, s', f, f).$$

Informally, $\mathsf{Map}_{\mathsf{end}}(T, T')$ is the type of maps that respect the endomorphism, and $\mathsf{Map}_{\mathsf{biInv}}(T, T')$ is the type of maps that respect the endomorphism and the proof that the endomorphism is a bi-invertible map.

We have a forgetful map $\mathsf{Map}_{\mathsf{biInv}}(T, T') \to \mathsf{Map}_{\mathsf{end}}(T, T')$, and what Lemma 2.8 says is that this map is an equivalence.

There is something special about the type family $\mathsf{isBiInv} : (A \to B) \to \mathcal{U}$, and that is that it is valued in propositions. One might wonder if Lemma 2.8 is a general principle, in the following sense. Say that we have a signature S for a type of algebras, and we extend it to a signature S', such that the fields we added take values in propositions. In the above example S corresponds to $(T : \mathcal{U})(f : T \to T)$ and S' corresponds to the extension $(T : \mathcal{U})(f : T \to T)(\phi : \mathsf{isBiInv}(s))$. As usual, given S'-algebras T, T', we have a forgetful map $\mathsf{Map}_{S'}(T, T') \to \mathsf{Map}_S(T, T')$. Is this map an equivalence in general?

The following example, suggested by Paolo Capriotti, shows that this is not necessarily the case.

Example 6.1. Consider S, the signature $(T : \mathcal{U})(o : T \to T \to T)$, and S', the extension

$$(T : \mathcal{U})(o : T \to T \to T)(tr : \mathsf{isSet}(T))$$
$$(e : T)\,(u : (t : T) \to o(t, e) = t \times o(e, t) = t)\,.$$

The S-algebras are the types with a binary operation, and the S'-algebras are the sets with a binary operation with a distinguished element that is a left and right unit.

The extension S' is propositional. This is because being a set is a proposition ([18, Theorem 7.1.10]), so tr inhabits a proposition, two left and right units must necessarily coincide, so e inhabits a proposition (assuming u), and the identity types of a set are propositions and these are closed under pi-types ([18, Theorem 7.1.9]), so u inhabits a proposition (assuming tr).

Let us see that for S'-algebras T, T' the forgetful map $\mathsf{Map}_{S'}(T, T') \to \mathsf{Map}_S(T, T')$ is not an equivalence in general. Let T be $(\mathbb{N}, +, \phi, 0, \psi)$, where ϕ is a proof that the natural numbers form a set, and ψ is a proof that 0 is a left and right unit for $+$. Let T' be $(\mathsf{bool}, \vee, \phi', \bot, \psi')$, where ϕ' is a proof that the booleans form a set, and ϕ is a proof that $\bot$ is a left

and right unit for $\vee$. Then we have $\lambda n.\top : \mathbb{N} \to \text{bool}$. This map clearly respects the operations, so we get an inhabitant of $\text{Map}_S(T, T')$. But this morphism does not respect the units, so it cannot come from a morphism in $\text{Map}_{S'}(T, T')$.

This discussion leaves open an interesting question.

Question 1. *Given a signature S and a propositional extension S', are there useful necessary and sufficient conditions for the forgetful map $\text{Map}_{S'}(T, T') \to \text{Map}_S(T, T')$ to be an equivalence for every pair of S'-algebras T and T'?*

Initiality of HIITs

Our original goal was to complete the conjectured result from [4] and formally verify that $\mathbb{Z}_h$ is a set. Using the strategy from this paper this is fairly straightforward: we can show that the natural notion of morphism of $\mathbb{Z}_h$-algebras satisfies a principle analogous to Lemma 2.8, and hence that $\mathbb{Z}_h$ is a set. When attempting to formalize this construction we hit an unexpected problem: it turns out that it is rather difficult to verify that the higher inductive type defining $\mathbb{Z}_h$ is initial in its corresponding wild category of algebras. Specifically, the proof seems to require the construction of a filler for a 4-dimensional cube which is rather laborious. In [12] it is shown that for QIITs (i.e., set-truncated HIITs) elimination and initiality are equivalent, but the extension to higher dimensional HIITs seems non-trivial. In particular it may require developing the higher order categorical structure of the category of algebras.

References

[1] 2019. RedTT. https://github.com/RedPRL/redtt.

[2] Andreas Abel, Anders Mörtberg, and Andrea Vezzosi. 2019. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *To appear in ICFP* (2019).

[3] Thorsten Altenkirch and Ambrus Kaposi. 2016. Type Theory in Type Theory Using Quotient Inductive Types. *SIGPLAN Not.* 51, 1 (Jan. 2016), 18–29. https://doi.org/10.1145/2914770.2837638

[4] Thorsten Altenkirch and Gun Pinyo. 2018. Integers as a Higher Inductive Type. *TYPES* (2018).

[5] Thorsten Altenkirch and Luis Scoccola. 2019. Implementation of The Integers as a Higher Inductive Type. https://github.com/LuisScoccola/integers-as-HITs.

[6] Henning Basold, Herman Geuvers, and Niels van der Weide. 2017. Higher Inductive Types in Programming. *J. UCS* 23 (2017), 63–88.

[7] Evan Cavallo. 2018. biinv-int. https://github.com/RedPRL/redtt/blob/master/library/cool/biinv-int.red.

[8] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015) (Leibniz International Proceedings in Informatics (LIPIcs))*, Tarmo Uustalu (Ed.), Vol. 69. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 5:1–5:34. https://doi.org/10.4230/LIPIcs.TYPES.2015.5

[9] Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '18)*. ACM, New York, NY, USA, 255–264. https://doi.org/10.1145/3209108.3209197

[10] Ambrus Kaposi. 2019. Separate definition of constructors? Agda mailing ist.

[11] Ambrus Kaposi and András Kovács. 2019. Signatures and Induction Principles for Higher Inductive-Inductive Types. *CoRR* abs/1902.00297 (2019). arXiv:1902.00297 http://arxiv.org/abs/1902.00297

[12] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing quotient inductive-inductive types. *PACMPL* 3, POPL (2019), 2:1–2:24. https://doi.org/10.1145/3290315

[13] Nicolai Kraus and Thorsten Altenkirch. 2018. Free Higher Groups in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. 599–608. https://doi.org/10.1145/3209108.3209183

[14] Nicolai Kraus and Jacob von Raumer. 2019. Path Spaces of Higher Inductive Types in Homotopy Type Theory. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–13.

[15] Nuo Li. 2015. *Quotient types in type theory*. Ph.D. Dissertation. University of Nottingham.

[16] D. R. Licata and G. Brunerie. 2015. A Cubical Approach to Synthetic Homotopy Theory. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. 92–103.

[17] Daniel R. Licata and Michael Shulman. 2013. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. *Proceedings - Symposium on Logic in Computer Science*, 223–232. https://doi.org/10.1109/LICS.2013.28

[18] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, Princeton, NJ. http://homotopytypetheory.org/book

Acknowledgments

The first author would like to thank Paolo Capriotti, Nicolai Kraus and Gun Pinyo for many interesting discussions on the subject of this paper. Both authors would like to thank Christian Sattler for comments and useful discussions. The work by and with Ambrus Kaposi and András Kovács plays an important role in particular in connection with the open questions triggered by this paper.