Natural numbers from integers

ABSTRACT

In homotopy type theory, a natural number type is freely generated by an element and an endomorphism. Similarly, an integer type is freely generated by an element and an automorphism. Using only dependent sums, identity types, extensional dependent products, and a type of two elements with large elimination, we construct a natural number type from an integer type. As a corollary, homotopy type theory with only Σ, Id, Π, and finite colimits with descent (and no universes) admits a natural number type. This improves and simplifies a result by Rose and settles a long-standing open question.

ACM Reference Format:
. 2024. Natural numbers from integers. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In standard set theory and other impredicative background theories, an object of natural numbers is easily constructed from some source of infinity. As soon as we have a set X with a "zero" element $z : X$ and an injective "successor" function $s : X \to X$ whose image does not contain z, we may carve out the natural numbers by taking the intersection of all subsets of X closed under z and s. However, this reasoning is essentially *impredicative*: for this definition to make sense (and yield the expected induction principle), we need the collection of all subsets of X to form a set, the power set of X.

In predicative settings such as Martin-Löf dependent type theory, this reasoning is ill-founded. Inductive type formers such as natural numbers cannot be reduced to non-inductive constructions: without impredicativity as a free source of induction principles, we are stuck. Therefore, the natural numbers (and other inductive types) are usually assumed axiomatically.

Homotopy type theory is a version of Martin-Löf type theory with function extensionality, univalence for any assumed universes, and generalizations of inductive types called *higher inductive types*. An example of a non-recursive higher inductive type is the *circle* S^1, freely generated by an element b and an identification of b with itself. As proved in [7], the loop space of the circle (the type of self-identifications of its base point) has the universal properties of the *integers*: it is freely generated by an element (given by the trivial self-identification) and an automorphism (given by postcomposition with the generating loop). Although its loop space has infinitely many elements, the type itself is generated using purely finite means: the higher inductive type of the circle does not have any recursive constructors. Categorically, it arises from only finite limits and colimits.

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The integers are certainly a source of infinity. And if our theory was impredicative, it is reasonable to expect that we can carve out a natural number type from it in some way. But in a predicative setting such as homotopy type theory, the problem of constructing the natural numbers from the integers is perplexingly challenging. For example, there does not seem to be an easy way of even defining the subtype of non-negative integers: all the standard approaches use an inductively defined predicate. The only possible form of induction, integer induction, is *reversible*, in the sense that the induction step must be an equivalence instead of an implication. That seems to be severe restriction.

The question to settle is this: in a predicative theory such as homotopy type theory, can we construct the natural numbers from the integers? To our knowledge, this question was first raised in a discussion between Egbert Rijke and Mike Shulman on the nForum [10]. (Instead of the integers, we may also assume non-recursive higher inductive types such as the circle and assume a univalent universe.)

To the surprise of many experts, Robert Rose settled this question positively in his PhD thesis [12]. His construction is quite complicated, reflected in the length of his thesis. There were also some caveats: he needs two nested univalent universes (although the outer one may be replaced by large elimination).

Our contribution in this paper is to provide a new construction of the natural numbers from the integers that we believe to be much simpler. Furthermore, our construction is more general: we do not need any univalent universes whatsoever. Instead, we just rely on effectivity of finite coproducts (that the coprojections are disjoint embeddings; equivalently, that we have a two-element type that satisfies descent, a version of large elimination that computes up to equivalence). In fact, we two construction: a direct one in Section 6 and an indirect one in Section 7.

The key novel idea of our work is the following. We would like to say that a natural number is a non-negative integer. So we would like to be able to define the map $\sigma : \mathbb{Z} \to \mathbf{2}$ from integers to Booleans that tells whether an integer is negative or non-negative. A priori it is unclear how to do this: our only tool is integer induction, but σ is not induced by any automorphism of $\mathbf{2}$. Consider instead how to define σ *supposing* we have the naturals $\mathbb{N}$ available: then we define an automorphism of the coproduct $\mathbb{N} \sqcup \mathbf{1} \sqcup \mathbb{N}$, get an induced map $\sigma : \mathbb{Z} \to \mathbb{N} \sqcup \mathbf{1} \sqcup \mathbb{N}$ (which happens to be an equivalence), and postcompose with the natural map $\mathbb{N} \sqcup (\mathbf{1} \sqcup \mathbb{N}) \to \mathbf{2}$. In our setting, we cannot use $\mathbb{N}$ but we can simply replace $\mathbb{N} \sqcup \mathbf{1} \sqcup \mathbb{N}$ with another coproduct $A \sqcup B \sqcup A$. This has an automorphism with the desired properties whenever we have an equivalence $A \simeq B \sqcup A$. In particular, one can define an equivalence $\mathbb{Z} \simeq \mathbb{Z} \sqcup \mathbb{Z}$, decomposing $\mathbb{Z}$ into an even part and an odd part, directly using integer induction.

The categorical analogue of our result is the following. Suppose we have a locally cartesian closed (higher) category with finite coproducts that satisfy descent. Then given an integer object, we have a natural number object. In particular, any locally cartesian closed (higher) category with finite colimits that satisfy descent has a natural number object. In the absence of an internal language result of our type theory in such higher categories (related conjectures are made in [5] with progress in [6, 8]), we believe that our constructions are sufficiently abstract that they can be directly replayed in any concrete model for higher categories.[1]

2 SETTING: A RESTRICTED TYPE THEORY

We work in dependent type theory with only a limited set of type formers as set out below. We use $\equiv$ to denote judgmental equality and $\equiv_{\text{def}}$ to denote judgmental definitions. We generally follow the homotopy type theory book [13] for notational conventions.

We have the identity type $x =_A y$ for $x, y : A$ in the sense of Martin-Löf (no equality reflection or axiom K). We have the unit type $\mathbf{1}$, dependent sum types $\sum_{x:A} B(x)$, and dependent product types $\prod_{x:A} B(x)$, all satisfying judgmental β- and η-laws. Dependent products are extensional, meaning function extensionality holds (phrased using identity types).

We assume a *two-element type* $\mathbf{2}$, freely generated by elements $0, 1 : \mathbf{2}$. Its universal property says that for any type C with elements $d_0, d_1 : C$, the type of maps $f : \mathbf{2} \to C$ with identifications $f(0) =_C d_0$ and $f(1) =_C d_1$ is contractible. From this, we can derive a dependent elimination principle whose computation rule holds up to identity type.

2.1 Descent

We assume the two-element type satisfies *descent*. This encapsulates elimination into a univalent universe, allowing us to omit universes from our type theory. The principle is as follows. Given types A_0 and A_1, we have a family $C(x)$ over $x : \mathbf{2}$ together with equivalences $C(0) \simeq A_0$ and $C(1) \simeq A_1$. Here, the notion of equivalence is defined as usual in homotopy type theory (using dependent sums and products). This is a homotopically weakened version of large elimination.

2.2 Basic consequences

Elimination principle for two-element type. From the universal property of the two-element type $\mathbf{2}$, we can derive a dependent elimination principle whose reduction rule holds up to identity type. More precisely, given $C(x)$ for $x : \mathbf{2}$ with $d_0 : C(0)$ and $d_1 : C(1)$, we obtain $e(x) : C(x)$ for $x : \mathbf{2}$ together with identifications $e(0) =_C d_0$ and $e(1) =_C d_1$. Moreover, the type of such e is contractible (this uses function extensionality).

Empty type. We may construct the empty type as

$$\mathbf{0} \equiv_{\text{def}} 0 =_{\mathbf{2}} 1.$$

Its recursion principle follows from descent for the two-element type: given any type A, we have a family $C(x)$ over $x : \mathbf{2}$ with $C(0) \simeq \mathbf{1}$ and $C(1) \simeq A$. From $y : 0 =_{\mathbf{2}} 1$, we then get $C(0) \simeq C(1)$, so $A \simeq \mathbf{1}$, meaning that A is contractible. From this, we obtain the elimination principle for $\mathbf{0}$ as usual.

[1]We note that a similar-looking result is claimed in Rasekh [9]. However, this is in a setting with impredicativity, which as explained above simplifies the problem greatly. Furthermore, we have found severe mistakes in that paper, for example in the proof of Lemma 1.2.1: $\mathrm{Eq}(g_1, g_2)$ is not generally a subspace of $\mathrm{Map}_{/X}(g_1, g_2)$, which seems to deal a blow to the approach. We have notified the author in December 2022. He agrees with the problem and we are waiting for a correction to be issued. We leave it to the LICS editor to decide how we should handle this citation.

$\mathbf{2}$ is a set. By descent, we have a family $C(x)$ over $x : \mathbf{2}$ with $C(0) \simeq \mathbf{1}$ and $C(1) \simeq \mathbf{0}$. The sum over $x : \mathbf{2}$ of $C(x)$ has an element at $x \equiv 0$. By induction over $\mathbf{2}$ and $\mathbf{0}$, every element in the sum is equal to it. This shows that $\sum_{x:\mathbf{2}} C(x)$ is contractible. Since $C(0)$ is contractible, it follows that $0 =_{\mathbf{2}} 0$ is contractible. A dual argument shows that $1 =_{\mathbf{2}} 1$ is contractible, making $\mathbf{2}$ a set by $\mathbf{2}$-induction.

Binary coproducts. From descent for the two-element type, we can construct the *binary coproduct type* $A_0 \sqcup A_1$ of types A_0 and A_1. First, we obtain $C(x)$ over $x : \mathbf{2}$ from descent. We then define

$$A_0 \sqcup A_1 \equiv_{\text{def}} \sum_{x:\mathbf{2}} C(x).$$

The coprojections $\tau_i : A_i \to A_0 \sqcup A_1$ for $i \in \{0, 1\}$ are defined by passing backward along the equivalence $D(i) \simeq A_i$. The universal property of binary coproducts reduces, using dependent products, to the universal property of the two-element type. Given functions $f_i : A_i \to D$ for $i \in \{0, 1\}$, we write $[f_0, f_1] : A_0 \sqcup A_1 \to D$ for the map given by the universal property. Equivalently, the dependent elimination principle for coproducts is justified by that principle for the two-element type. Its reduction rules again hold up to identity type. We use the notation $[-, -]$ also in this dependent case.

No confusion for binary coproducts. The constructors for binary coproducts are disjoint embeddings. Disjointness follows from the tautology $\neg(0 =_{\mathbf{2}} 1)$. The embedding property reduces to contractibility of $0 =_{\mathbf{2}} 0$ and $1 =_{\mathbf{2}} 1$.

Descent for binary coproducts. Descent for the two-element type implies descent for binary coproducts. No confusion implies descent for binary products (in fact, it is equivalent to our assumption of descent for the two-element type). This is the following principle, again encapsulating a homotopically weakened version of large elimination into a univalent universe. Given families $B_i(a_i)$ over $a_i : A_i$ for $i \in \{0, 1\}$, we have a family $D(y)$ over $y : A_0 \sqcup A_1$ with equivalences $D(\tau_i(a_i)) \simeq B_i(a_i)$. To justify it, we first construct

$$D' \equiv_{\text{def}} \Big(\sum_{a_0:A_0} B_0(a_0)\Big) \sqcup \Big(\sum_{a_1:A_1} B_1(a_1)\Big).$$

Coproduct recursion induces a map $p : D' \to A_0 \sqcup A_1$. By no confusion, this map pulls back along τ_i for $i \in \{0, 1\}$ to the projection from $\sum_{a_i:A_i} B_i(a_o)$ to A_i. We may thus define $D(y)$ as the fiber of p over y.

Finite coproducts with no confusion and descent. We derive finite coproducts from binary coproducts and the empty type. This extends to properties such as no-confusion or descent (note that descent for nullary coproducts is vacuous). For a coproduct of (external) arity $k \in \mathbb{N}$, we denote the coprojections by τ_i where $0 \le i < k$.

2.3 Formalization

The first approach in this paper has been formalized in Agda [2]. Where appropriate, lemmas and theorem in our paper come with a reference to the corresponding part of the formalization. Agda has many features beyond the setting of this paper, including universes and general inductive types. The formalization avoids such features and does not use any library. The only inductive types declared in the formalization are the ones in our setting. It does not seem feasible to avoid mentioning the first universe Set entirely when formalizing Agda, since it is used for example for type polymorphism and to express the judgement "A is a type", but we restrict our uses of Set to such places where it could in principle be eliminated.

Isomorphisms and equivalences. A fundamental result in any library of homotopy type theory is the fact that every map with a two-sided inverse has contractible fibers, and we need this result also in the present work. The well-known proof of this result, reproduced for example in Section 10.4 of [11], is not entirely trivial, requiring some higher path algebra. We present a new proof which we argue is simpler and easier to remember. A similar argument appears in lectures notes by Martín Escardó [3].

We start from two types A and B with maps $f : A \to B$ and $g : B \to A$ such that $g(f(a)) = a$ for all $a : A$ and $f(g(b)) = b$ for all $b : B$. We claim that the action on paths

$$\mathsf{ap}_f : (a =_A a') \to (f(a) =_B f(a'))$$

is an isomorphism for all $a, a' : A$, i.e. it also has a left inverse and a right inverse. We have a map

$$\mathsf{ap}_g : (f(a) =_B f(a')) \to (g(f(a)) =_A g(f(a'))$$

and an isomorphism

$$e : (g(f(a)) =_A g(f(a')) \to (a =_A a')$$

since $g(f(a)) = a$ and $g(f(a')) = a'$. We have

$$e(\mathsf{ap}_g(\mathsf{ap}_f(p))) =_{a=_A a'} p$$

for $p : a =_A a'$ by path induction on p and using that e sends reflexivity to reflexivity. Thus ap_f has a left inverse. By the same argument, swapping the roles of f and g, ap_g has a left inverse. Since e is an isomorphism, ap_g has a left inverse and a right inverse, so it is an isomorphism. Thus ap_f is also an isomorphism. From this, we prove that f has contractible fibers as follows. Since $b = f(g(b))$ for $b : B$, it suffices to consider the fiber of f over $f(a)$ for $a : A$, that is $\sum_{a':A} f(a) =_B f(a')$. Since ap_f is an isomorphism, this is isomorphic to $\sum_{a':A} a =_A a'$, a singleton, hence contractible.

3 NATURAL NUMBERS AND INTEGERS

A *natural number algebra* is a type A together with an element $z : A$ and an endofunction $s : A \to A$. Note that this is an external notion (lacking universes, we may not quantify internally over types A in our type theory). We generally refer to a natural number algebra just by its underlying type. We use z and s generically to refer to the structure components of a natural number algebra A. The *structure map* of A is the map $\mathbf{1} \sqcup A \to A$ induced by z and s.

Given natural number algebras A and B, the type of *algebra morphisms* from A to B consists of a function $f : A \to B$ with $f(z) =_B z$ and $f(s(a)) =_B s(f(a))$. We call a natural number algebra A *initial* if this type is contractible for every B. We then say A is a *natural number type*. Every natural number algebra A has an identity algebra morphism id_A. Algebra morphisms $f : A \to B$ and $g : B \to C$ admit a composition $g \circ f : A \to C$. These operations satisfy unit and associativity laws (phrased using identity types). We will not need any higher coherence conditions.

We also have displayed analogues of these notions. Given a natural number A, a *natural number algebra B displayed over A* is a family $B(a)$ over $a : A$ with $z : B(z)$ and $s_a(b) : B(s(a))$ for $a : A$ and $b : B(a)$. Its *total algebra* is obtained by taking the dependent sum. The type of *sections* of B consists of $b(a) : B(a)$ for $a : A$ together with $b(z) =_{B(z)} z$ and $b(s(a)) =_{B(s(a))} s(b(a))$ for $a : A$. We say that A has *elimination* if every natural number algebra displayed over it has a section. As is standard, one proves (externally):

Lemma 3.1. *A natural number algebra is initial exactly if it has elimination.* □

The following notion features in our second approach. A variation with a family of nullary constructors appears in our first approach.

Definition 3.2. A natural number algebra A is *stable* if its structure map $[z, s] : \mathbf{1} \sqcup A \to A$ is an equivalence.

This means that $z : \mathbf{1} \to A$ and $s : A \to A$ form a colimiting cocone, i.e., that A is *non-recursively* freely generated by z and s. Lambek's lemma states that every initial natural number algebra is stable.

We call a natural number algebra A an *integer algebra* if its endofunction s is an equivalence. Note that this condition is a proposition. This justifies defining morphisms of integer algebras as morphisms of the underlying natural number algebras. An integer algebra A is *initial* if the type of integer algebra morphisms from A to B is contractible for any integer algebra B. We then say that A is an *integer type* (more verbosely, a *type of integers*).

We have a notion of displayed integer algebra analogous to the case of natural number algebras. The type of sections of a displayed integer algebra is defined as the type of sections of the underlying displayed natural number algebra. Analogous to the case of natural numbers, we have:

Lemma 3.3. *An integer algebra is initial exactly if it has elimination.* □

Our goal in the rest of this article is to construct a natural number type from an integer type. We thus now make the standing assumption of an integer type $\mathbb{Z}$, with element denoted $Z : \mathbb{Z}$ and automorphism denoted $S : \mathbb{Z} \simeq \mathbb{Z}$ (to distinguish from the natural number algebras we will consider). Note that any other integer type is equivalent to it (by universality). It thus makes sense to speak of *the* integer type $\mathbb{Z}$.

The rest of this paper is devoted to proving the following:

Theorem 3.4. *Assume an integer type. Then we have a natural number type.*

We provide two separate proofs, a direct one in Section 6 and an indirect one in Section 7.

4 AN EQUIVALENCE $\mathbb{Z} \simeq \mathbb{Z} \sqcup \mathbb{Z}$

Our starting point for constructing the natural numbers is the following observation.

Lemma 4.1 (`Doubling.halve-iso`). *We have an equivalence $\mathbb{Z} \simeq \mathbb{Z} \sqcup \mathbb{Z}$.*

Proof. We first define the operation of *squaring* integer algebras. Given an integer algebra $X \equiv (X, z, s)$, its square $\mathrm{Sq}(X)$ is the integer algebra $(X, z, s \circ s)$. Note that $s \circ s$ is an equivalence since s is.

This operation is functorial: it has an evident action on morphisms of integer algebras (we do not need any higher witnesses of functoriality). Furthermore, the functorial action reflects equivalences: if $\mathrm{Sq}(f)$ is invertible, then so is f.

Next, for an integer algebra $X \equiv (X, z, s)$, we define the *twisted rotation* integer algebra $\mathrm{Tw}(X)$ with carrier $X \sqcup X$, element $\tau_0(z)$, and automorphism r mapping $\tau_0(x)$ to $\tau_1(x)$ and $\tau_1(x)$ to $\tau_0(s(x))$. This uses the universal property of binary coproducts to define r.

The automorphism of $\mathrm{Sq}(\mathrm{Tw}(X))$ maps $\tau_0(x)$ to $\tau_0(s(x))$ and $\tau_1(x)$ to $\tau_1(s(x))$. That is, on each component, it is just given by the original automorphism s. In particular, τ_0 forms an algebra morphism from X to $\mathrm{Sq}(\mathrm{Tw}(X))$.

By initiality of $\mathbb{Z}$, we have an algebra map $\mathbb{Z} \to \mathrm{Sq}(\mathbb{Z})$. Let us call its underlying map $\mathrm{double} : \mathbb{Z} \to \mathbb{Z}$. Note that

$$[\mathrm{double}, S \circ \mathrm{double}] : \mathbb{Z} \sqcup \mathbb{Z} \to \mathbb{Z}$$

forms an algebra map from $\mathrm{Tw}(\mathbb{Z})$ to $\mathbb{Z}$.

Again by initiality of $\mathbb{Z}$, we have an algebra map $c : \mathbb{Z} \to \mathrm{Tw}(\mathbb{Z})$. To finish the proof, it suffices to show that its underlying function $\mathbb{Z} \to \mathbb{Z} \sqcup \mathbb{Z}$ is invertible. By initiality of $\mathbb{Z}$, the algebra morphism composite $[\mathrm{double}, S \circ \mathrm{double}] \circ c$ is the identity, hence also on underlying functions. It remains to show that the underlying function of the algebra endomorphism

$$u \equiv_{\mathrm{def}} c \circ [\mathrm{double}, S \circ \mathrm{double}]$$

on $\mathrm{Tw}(\mathbb{Z})$ is the identity.

Note that $\mathrm{Sq}(u)$ is an algebra endomorphism on $\mathrm{Sq}(\mathrm{Tw}(\mathbb{Z}))$. By initiality of $\mathbb{Z}$, we have $\mathrm{Sq}(u) \circ \tau_0 = \tau_0$. On underlying maps, this means $u \circ \tau_0 = \tau_0$. It remains to check $u \circ \tau_1 = \tau_1$. Given $x : \mathbb{Z}$, we calculate

$$\begin{aligned} u(\tau_1(x)) &= c(S(\mathrm{double}(x))) \\ &= r(c(\mathrm{double}(x))) \\ &= r(u(\tau_0(x))) \\ &= r(\tau_0(x)) \\ &= \tau_1(x). \end{aligned}$$ □

Remark 4.1. We note an abstract perspective on Lemma 4.1. We have an endo-adjunction on the (higher) category of natural number algebras: the left adjoint is twisted rotation and the right adjoint is squaring. Both functors preserve integer algebras, so the adjunction descends to the full subcategory of integer algebras. As left adjoints preserve initial objects, the twisted rotation $\mathrm{Tw}(\mathbb{Z})$ is again initial, in particular equivalent to $\mathbb{Z}$.

To render this reasoning in our type theory without universes, we represent (higher) categories as a mixed external-internal notion: external at object level and internal at morphism level and above. That is, we have (external) *sets* of objects (since those involve types in the case of algebras), but *types* of morphisms. As usual, the construction depends only on a finite approximation of the higher-categorical coherence tower, in this case just composition of morphisms and one level of coherence. For the adjunction, we just need the equivalence between hom-types without naturality. The given proof of Lemma 4.1 can be seen as the unravelling of the abstract perspective in this manner.

(This mixed reasoning can be formalized in two-level type theory [1] using a variation of the notion of *wild category* with an outer type of objects and inner types of morphisms.)

5 APPROXIMATING THE INTEGERS VIA HALVES

If we already had the natural numbers $\mathbb{N}$, we could describe the integers as being built out of two copies of the natural numbers, one for the positive and one for the negative half:

$$\mathbb{Z} \simeq \mathbb{N} \sqcup \mathbf{1} \sqcup \mathbb{N} \tag{1}$$

Conversely, we may use a decomposition with similar properties to tell us something about the integers, for example when an integer is positive or negative. This is the motivation behind the following construction.

Construction 5.1 (`Signs.shift-equiv`). Consider types A and B with an equivalence $e : A \simeq B \sqcup A$. Then we have an automorphism on $A \sqcup B \sqcup A$ given by reassociating the equivalence

$$A \sqcup (B \sqcup A) \simeq (A \sqcup B) \sqcup A$$

where we act using e on the left component and using the inverse of e on the right component. Given also an element $b : B$, this forms an integer algebra structure on $A \sqcup B \sqcup A$.

In the above situation, initiality of $\mathbb{Z}$ provides us with an algebra morphism from $\mathbb{Z}$ to $A \sqcup B \sqcup A$. Restricting along the underlying function, the ternary decomposition on the right induces a decomposition

$$\mathbb{Z} \simeq \mathbb{Z}^- \sqcup \mathbb{Z}^0 \sqcup \mathbb{Z}^+ \tag{2}$$

of $\mathbb{Z}$ into three parts (this uses effectiveness of coproducts – that the coprojections are disjoint embeddings). The automorphism S of $\mathbb{Z}$ restricts to separate equivalences $S^- : \mathbb{Z}^- \simeq \mathbb{Z}^- \sqcup \mathbb{Z}^0$ and $S^+ : \mathbb{Z}^0 \sqcup \mathbb{Z}^+ \simeq \mathbb{Z}^+$ that combine to give S via the above decomposition. Furthermore, since $Z : \mathbb{Z}$ is sent to $\tau_1(b)$, we know that Z lies in the middle component $\mathbb{Z}^0$.

In the presence of the naturals, one may prove that the decomposition (2) in fact agrees with the decomposition (1).

6 FIRST APPROACH: DIRECT

We now give a direct construction of an initial natural number algebra. First we apply Construction 5.1 to the equivalence $\mathbb{Z} \simeq \mathbb{Z} \sqcup \mathbb{Z}$ from Lemma 4.1 and the element $Z : \mathbb{Z}$ to obtain the decomposition (2). We write M for $\mathbb{Z}^0 \sqcup \mathbb{Z}^+$. By construction, for $x : \mathbb{Z}$, we have that x lies in M iff $S(x)$ lies in $\mathbb{Z}^+$, so S restricts to an equivalence $M \simeq \mathbb{Z}^+$. Thus we get also an equivalence $\mathbb{Z}^0 \sqcup M \simeq M$. We write S also for the induced map $M \to M$ lying above $S : \mathbb{Z} \to \mathbb{Z}$.

We will show that M has a universal property close to that of the natural numbers: it is freely generated by $\mathbb{Z}^0 \to M$ and $S : M \to M$. Note also that we have an element of $\mathbb{Z}^0$, namely Z. To construct $\mathbb{N}$ from here, we will use a simple rectification argument.

We first explain how to prove that M has the stated universal property. We essentially follow a well-known strategy for reducing natural number recursion (which constructs *functions* out of $\mathbb{N}$) to natural number induction (which proves proposition-valued *predicates* on $\mathbb{N}$) by considering an appropriate notion of "partially defined inductive function". To this end, we first need a notion of ordering on $\mathbb{Z}$.

Lemma 6.1 (`Signs.agda`). *$\mathbb{Z}$ has a proposition-valued relation $<$ with the following properties:*

(i) if $x < y$, then $x < S(y)$,
(ii) if $S(x) < S(y)$, then $x < y$,
(iii) if $x : M$ then we do not have $x < Z$,
(iv) $x < S(x)$ for all x.

Proof. Using initiality of $\mathbb{Z}$, we can define subtraction on $\mathbb{Z}$ such that $x - Z = x$ and $x - S(y) = S^{-1}(x - y)$. It can then be proven by integer induction that $S(x) - S(y) = x - y$ and $x - x = Z$. We take $x < y$ to mean that $x - y$ lies in $\mathbb{Z}^-$. All the listed properties can be verified directly. □

We will suppress witnesses of the relation $<$, writing a dash in their place, relying on references to the previous lemma to fill them as required. This is harmless as $<$ is valued in propositions.

We recall some preliminaries on fixpoints. Given an endofunction t on a type X, we write $\mathsf{fix}(t)$ for the type of *fixpoints* of t:

$$\mathsf{fix}(t) \equiv_{\mathrm{def}} \sum_{x:X} t(x) = x.$$

The below "rolling rule" is useful for manipulating fixpoints.

Lemma 6.2 (`RollingRule.agda`). *For $f : X \to Y$ and $g : Y \to X$, we have an equivalence $\mathsf{fix}(g \circ f) \simeq \mathsf{fix}(f \circ g)$.*

Proof. Both types arise from

$$\sum_{x:X} \sum_{y:Y} (f(x) = y) \times (g(y) = x)$$

by contracting singletons. □

Note that the forward map $\mathsf{fix}(g \circ f) \to \mathsf{fix}(f \circ g)$ is f applied to the first component, and the first component of the inverse map is similarly given by g. This follows from unfolding the proof above.

6.1 The universal property of M

This subsection is devoted to proving the universal property of M:

Proposition 6.3 (`M.M-ind`). *Let A be a family over M together with*

- *$z_A(x) : A(x)$ for $x : \mathbb{Z}^0$,*
- *$s_A(a) : A(S(x))$ for (implicit) $x : M$ with $a : A(x)$.*

Then we have $g(x) : A(x)$ for $x : M$ such that:

- *$g(x) = z_A(x)$ for $x : \mathbb{Z}^0$,*
- *$g(S(x)) = s_A(g(x))$ for $x : M$.*

Throughout this subsection, we fix A together with z_A and s_A as above. We define for $u : \mathbb{Z}$ a type

$$\mathsf{pfun}(u) \equiv_{\mathrm{def}} \prod_{x:M} (x < u) \to A(x)$$

of partial sections of A defined below u. Intuitively, an element of $\mathsf{pfun}(u)$ is a section of A defined on a finite prefix of M. For $u : \mathbb{Z}$, we have a *restriction* map

$$\mathsf{res}_u : \mathsf{pfun}(S(u)) \to \mathsf{pfun}(u)$$

using part (i) of Lemma 6.1 and an *extension* map

$$\mathsf{ext}_u : \mathsf{pfun}(u) \to \mathsf{pfun}(S(u))$$

given by a case distinction using the equivalence $\mathbb{Z}^0 \sqcup M \simeq M$:

- $\mathsf{ext}_u(f, x, -) \equiv_{\mathrm{def}} z_A(x)$ for x in $\mathbb{Z}^0$,
- $\mathsf{ext}_u(f, S(x), -) = s_A(f(x, -))$ using part (ii) of Lemma 6.1.

Lemma 6.4 (`ext-res-eq-res-ext`). *The operations* res *and* ext *commute in the sense that $\mathsf{ext}_u \circ \mathsf{res}_u = \mathsf{res}_{S(u)} \circ \mathsf{ext}_{S(u)}$ for $u : \mathbb{Z}$.*

Proof. For each $f : \mathsf{pfun}(S(u))$ and $x : M$ with $x < u$, we have to prove an equality in $A(x)$. We do a case distinction on x using the equivalence $\mathbb{Z}^0 \sqcup M \simeq M$. Each case is direct by unfolding definitions. □

Now we define for $u : \mathbb{Z}$ a type

$$\mathsf{indfun}(u) \equiv_{\mathrm{def}} \mathsf{fix}(\mathsf{res}_u \circ \mathsf{ext}_u)$$

of partial sections f of A defined below u together with a witness that f is *inductive*. To understand this terminology, note that $\mathsf{res}_u(\mathsf{ext}_u(f)) = f$ unfolds to the recursive equation $f(x, -) = z_A(x)$ for $x : \mathbb{Z}^0$ with $x < u$ and $f(S(x), -) = s_A(f(x), -)$ for $x : M$ with $S(x) < u$. We defined indfun in this more compact way in order to obtain a simple proof of the following result.

Lemma 6.5 (`M.M-ind-aux`). *For all $u : \mathbb{Z}$, we have an element $f(u)$ of $\mathsf{indfun}(u)$.*

Proof. By integer induction. We trivially have an element of $\mathsf{indfun}(Z)$, since there is no $x : M$ with $x < Z$. Moreover, we have

$$\begin{aligned}\mathsf{indfun}(S(u)) &\simeq \mathsf{fix}(\mathsf{res}_{S(u)} \circ \mathsf{ext}_{S(u)}) \\ &\simeq \mathsf{fix}(\mathsf{ext}_u \circ \mathsf{res}_u) \\ &\simeq \mathsf{fix}(\mathsf{res}_u \circ \mathsf{ext}_u) \\ &\simeq \mathsf{indfun}(u)\end{aligned}$$

by Lemma 6.4 and the rolling rule. □

In fact, one can strengthen the above result to the claim that $\mathsf{indfun}(u)$ is contractible – so in particular the proof only uses integer induction for propositions – but we will not need this strengthening.

Proof of Proposition 6.3. We define $g(x)$ as the evaluation of $f(S(x))$ at x, using part (iv) of Lemma 6.1. The first equation follows from the fact that $f(S(Z))$ is inductive. The second equation follows from the fact that $f(S(x)) = \mathsf{ext}_x(f(x))$, which we have by construction of f and the fact that the rolling map

$$\mathsf{fix}(\mathsf{res}_x \circ \mathsf{ext}_x) \to \mathsf{fix}(\mathsf{ext}_x \circ \mathsf{res}_x)$$

is $\mathsf{ext}_x : \mathsf{indfun}(x) \to \mathsf{indfun}(S(x))$ on first components. □

6.2 Defining the natural numbers

Lemma 6.6 (`Naturals.agda`). *Let X and Y be types with maps $\iota : Y \to X$ and $s : X \to X$. Suppose that X is freely generated by these two maps. If Y has an element, then we have a natural number type.*

Proof. Let $z : Y$ be given. We define a self-map $r : X \to X$ by $r(\iota(y)) = \iota(z)$ and $r(s(x)) = s(r(x))$ using the universal property of X. Let $\mathbb{N} \equiv_{\mathrm{def}} \sum_{x:X} r(x) = x$ be the type of fixpoints of r. By Lambek's lemma,[2] the map $Y + X \to X$ is an equivalence, so in particular ι and s are embeddings. We have an equivalence

[2] In the case at hand, we start from an equivalence $\mathbb{Z}^0 \sqcup M \simeq M$ and there is no need to invoke Lambek's lemma.

$e_\iota : (z = y) \simeq (r(\iota(y)) = \iota(y))$ since ι is an embedding, and $e_s : (r(x) = x)) \simeq (r(s(x)) = s(x))$ since s is an embedding. Thus we have an element $z_{\mathbb{N}} : \mathbb{N}$ given by $(z, e_\iota(\mathsf{refl}))$. We also have an endomorphism $s_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ given by $s_{\mathbb{N}}(x, p) = (s(x), e_s(p))$.

We claim that this makes $\mathbb{N}$ a natural number type. Thus let P be a type family over $\mathbb{N}$ with $z_P : P(z_{\mathbb{N}})$ and $s_P : \prod_{n:\mathbb{N}} P(n) \to P(s_{\mathbb{N}}(n))$. We construct an element $p : \prod_{x:X} \prod_{h:r(x)=x} P(x, h)$ using the universal property of X. We first need to construct, for $y : Y$, an element $\prod_{h:r(\iota(y))=\iota(y)} P(y, h)$, or equivalently $\prod_{h:z=y} P(y, e_\iota(h))$. We can define this by path induction using z_P. Then we have to construct, for $x : X$ and $f : \prod_{h:r(x)=x} P(x, h)$, an element of $\prod_{h:r(s(x))=s(x)} P(s(x), h)$, or equivalently $\prod_{h:r(x)=x} P(s(x), e_s(h))$. Given $h : r(x) = x$, we simply use $s_P(x, h)(f(h))$. It is direct to verify that this defines a section of P as a displayed natural numbers algebra. □

Proof of Theorem 3.4. Apply Lemma 6.6 to M with its universal property (Proposition 6.3). □

7 SECOND APPROACH: INDIRECT

The strategy of our second approach is more indirect and perhaps more in the spirit of Rose [12]. The key operation is *stabilizing* natural number algebras: carving out a fragment (not generally a subtype) on which the structure map is invertible. We achieve this by storing with each element a trace of previous elements that explains how the element is obtained by iterations of the structure map. To construct a type of such traces for an element x, we need a function type valued in the carrier of the algebra, but domain varying in x. If we had access to a univalent universe, we could make the choice of such a totally ordered domain type part of the data of the trace (we believe this is the essential part of Rose's construction). Without univalence, the extra redundancy of this choice creates problems. We eliminate this redundancy by forcing (the code for) the domain type to be uniquely determined by the values the trace function takes. The technical complexity of the development results from encoding this *very dependent type* [4]. Lastly, to avoid a universe entirely, we custom-build a universe out of fixpoints of endofunctions on the integers that is closed under sufficient type formers.

7.1 A custom universe

We say that a family of types $B(x)$ for $x : A$ has *binary coproducts* if, for $a_0, a_1 : A$, we have $t : A$ and maps $B(a_0) \to B(t)$ and $B(a_1) \to B(t)$ exhibiting $B(t)$ as a binary coproduct (equivalently, we have $t : A$ with an equivalence $B(t) \simeq B(a_0) \sqcup B(a_1)$). We similarly define when the family has *empty types (nullary coproducts)* and *unit types*. The intuition is that we see A as a universe and B as the associated universal family, sending a code in the universe to an actual type. Note that we do not assume that the family is univalent.

Lemma 7.1. *There is a family* El *over a type* U *that has unit types and finite coproducts.*

Proof. We take $U \equiv_{\mathsf{def}} \mathbb{Z} \to \mathbb{Z}$ and El as taking fixpoints:

$$\mathsf{El}(f) \equiv_{\mathsf{def}} \sum_{x:\mathbb{Z}} f(x) =_{\mathbb{Z}} x.$$

The unit type is coded by the function constant on Z. For finite coproducts, we make use of the equivalence $\mathbb{Z} \simeq \mathbb{Z} \sqcup \mathbb{Z}$ provided Lemma 4.1. Under this equivalence, it suffices to exhibit codes as fixpoints of endofunctions on $\mathbb{Z} \sqcup \mathbb{Z}$. Calculating the fixpoints of these endofunctions then makes use of the no-confusion property for binary coproducts.

- The empty type is coded by the endofunction on $\mathbb{Z} \sqcup \mathbb{Z}$ swapping the two components. The type of fixpoints of this endofunction is empty.
- The binary coproduct of $f, g : U$ is coded by the endofunction $f \sqcup g$ on $\mathbb{Z} \sqcup \mathbb{Z}$ that is separately f on the left component and g on the right component. Its type of fixpoints is equivalent to the coproduct of the types of fixpoints of f and g. □

7.2 Counting structures

Definition 7.2. The type of *successor structures* from a type C to a type D is the record type (iterated dependent sum) with the following data:

- $\mathsf{min} : D$ and $\mathsf{upp} : C \to D$, together freely generating,
- $\mathsf{low} : C \to D$ and $\mathsf{max} : D$, together freely generating,

Note that free generation is expressed by a propositional type. Equivalently, the maps

$$[\mathsf{min}, \mathsf{upp}] : \mathbf{1} \sqcup C \to D,$$
$$[\mathsf{low}, \mathsf{max}] : C \sqcup \mathbf{1} \to D$$

are invertible. The induced equivalence

$$\mathbf{1} \sqcup C \simeq D \simeq C \sqcup \mathbf{1}$$

can be thought of as a successor relation on C that misses a unique predecessor and successor from being an equivalence (as for example for a finite prefix of the natural numbers). We could strengthen the definition to work with decidable total orders and require that the above equivalences lifts to

$$\mathbf{1} \star C \simeq D \simeq C \star \mathbf{1}$$

where $A \star B$ denotes the *join* of orders A and B. However, we do not need this in our development.

Definition 7.3. A *counting structure* on a natural number algebra A is a family C over A with $\neg C(z)$ and a successor structure $(\mathsf{min}_x, \mathsf{upp}_x, \mathsf{low}_x, \mathsf{max}_x)$ from $C(x)$ to $C(s(x))$ for $x : A$ such that $\mathsf{min}_{s(x)} \neq \mathsf{max}_{s(x)}$ in $C(s(s(x)))$ for $x : A$. A *counting algebra* is a natural number algebra equipped with a counting structure.

The notion of counting structures is motivated by our desire to associate to each natural number a set of that cardinality.

Lemma 7.4. *There is a counting algebra.*

Proof. From Lemma 7.1, we have a family El over a type U with unit types and finite coproducts. For our counting algebra, we take as carrier the type of tuples (c, d, s) of $c, d : U$ with a successor structure $s = (\mathsf{min}, \mathsf{upp}, \mathsf{low}, \mathsf{max})$ from $\mathsf{El}(c)$ to $\mathsf{El}(d)$. The zero element has c given by the code for the empty type and d given by the code for the unit type, with essentially unique successor structure. The successor of (c, d, s) is given by (d, e, t) where e is a code for $\mathsf{El}(d) \sqcup \mathbf{1}$ and $t = (\mathsf{min}', \mathsf{upp}', \mathsf{low}', \mathsf{max}')$ has $\mathsf{low}' = \tau_0$

and $\mathsf{max}' = \tau_1(-)$ while $[\mathsf{min}', \mathsf{upp}']$ fills the below square of equivalences:

$$\begin{tikzcd}
\mathbf{1} \sqcup (\mathsf{El}(c) \sqcup \mathbf{1}) \arrow[r, "\simeq"] \arrow[d, "\mathbf{1}\sqcup[\mathsf{low},\mathsf{max}]"', "\simeq"] & (\mathbf{1} \sqcup \mathsf{El}(c)) \sqcup \mathbf{1} \arrow[d, "\simeq"', "{[\mathsf{min},\mathsf{upp}]\sqcup\mathbf{1}}"] \\
\mathbf{1} \sqcup \mathsf{El}(d) \arrow[r, dotted, "{[\mathsf{min}',\mathsf{upp}']}"] & \mathsf{El}(d) \sqcup \mathbf{1}.
\end{tikzcd}$$

Note that $\mathsf{min}' \neq \mathsf{max}'$ by effectivity of binary coproducts. □

Corollary 7.5. *Every natural number algebra receives a map from a counting algebra.*

Proof. Take the product of the given natural number algebra with the counting algebra of Lemma 7.4. Since counting structures are contravariant in the natural number algebra, this product inherits a counting structure. □

7.3 Stabilization

The following statement is our only use of counting structures. It provides a mechanism for annotating an algebra element with a "recursive" vector of algebra elements.

Lemma 7.6. *Every counting algebra A admits the following:*

- *a type $M(x)$ for $x : A$ with:*
  - *$M(z)$ is contractible,*
  - *an equivalence $\mathsf{pair} : M(x) \times A \to M(s(x))$,*
- *$\mathsf{last}_x(m) : \mathbf{1} \sqcup A$ for $m : M(x)$ with identifications:*
  - *$\mathsf{last}_z(-) = \tau_0(*)$,*
  - *$\mathsf{last}_{s(x)}(\mathsf{pair}(m, y)) = \tau_1(y)$ for $m : M(x)$ and $y : A$,*
- *$\mathsf{rebuild}_x(m) : M(x)$ for $m : M(x)$ with identification*

$$\mathsf{rebuild}_{s(x)}(\mathsf{pair}(m, y)) =_{M(s(x))} \mathsf{pair}(\mathsf{rebuild}_x(m), \mathsf{next}_x(m))$$

where $\mathsf{next}_x(m) \equiv_{\mathrm{def}} [z, s](\mathsf{last}_x(m)) : A$.

Proof. Let C denote the counting structure of A. We define

$$M(x) \equiv_{\mathrm{def}} C(x) \to A.$$

Note that $M(z)$ is contractible since $\neg C(z)$. The equivalence

$$\mathsf{pair} : (C(x) \to A) \times A \simeq (C(s(x)) \to A)$$

is induced by freeness of $\mathsf{low} : C(x) \to C(s(x))$ and $\mathsf{max} : C(s(x))$.

For the other data, we use that $C(s(x))$ is freely generated by min and upp. The element $\mathsf{last}_x(m) : \mathbf{1} \sqcup A$ is defined by case distinction on $\mathsf{max} : C(s(x))$:

- on min, we return $\tau_0(*)$,
- on $\mathsf{upp}(y)$ with $y : C(x)$, we return $\tau_1(m(y))$.

The element $\mathsf{rebuild}_x(m)(c) : A$ for $m : C(x) \to A$ and $c : C(x)$ is defined by case distinction on $\mathsf{low}(c) : C(s(x))$:

- on min, we return z,
- on $\mathsf{upp}(y)$ with $y : C(x)$, we return $m(y)$.

All the required identifications are direct. □

We now use the vectors provided by the previous lemma to annotate an algebra element with a trace witnessing that it is obtained recursively from the structure map. Restricting to elements with such a trace stabilizes the natural number algebra.

Lemma 7.7. *Every natural number algebra receives a map from a stable natural number algebra (see Definition 3.2).*

Proof. Let A denote the given natural number algebra. Using Corollary 7.5, we may reduce to the setting where A comes equipped with a counting structure. There, we have the structure given by Lemma 7.6.

We define a natural number algebra B with underlying type the following record (iterated dependent sum):

- $x : A$,
- $m : M(x)$,
- $q : x = \mathsf{next}_x(m)$.
- $p : m =_{M(x)} \mathsf{rebuild}_x(m)$,

This lies over A via the first projection. Given $x : A$, we write $B(x)$ for the record of the remaining three components. It remains to construct an equivalence $\mathbf{1} \sqcup B \simeq B$ over $[z, s] : \mathbf{1} \sqcup A \to A$. We construct this equivalence in reverse direction as a series of steps.

First, B arises by contracting k and α in the following record:

- $x : A$,
- $k : \mathbf{1} \sqcup A$,
- $q : x =_A [z, s](k)$,
- $m : M(x)$,
- $\alpha : k =_{\mathbf{1}\sqcup A} \mathsf{last}_x(m)$,
- $p : m =_{M(x)} \mathsf{rebuild}_x(m)$.

Contracting x with q, we obtain the equivalent record:

- $k : \mathbf{1} \sqcup A$,
- $m : M(x)$,
- $\alpha : k = \mathsf{last}_x(m)$,
- $p : m = \mathsf{rebuild}_x(m)$

where the reverse direction sets $x \equiv_{\mathrm{def}} [z, s](k)$ as required. It remains to show that the record of the last three fields is equivalent to $B(k)$. For this, we perform case distinction on k.

For $k = \tau_0(*)$, we have $x = z$ and are equivalently left with:

- $m : M(z)$,
- $\alpha : \tau_0(*) = \mathsf{last}_z(m)$,
- $p : m = \mathsf{rebuild}_z(m)$.

Since $M(z)$ is contractible, this is equivalent to $\tau_0(*) =_{\mathbf{1}\sqcup A} \tau_0(*)$. By no-confusion, τ_0 is an embedding, so this is contractible.

For $k = \tau_1(y)$, we have $x = s(y)$ and are equivalently left with:

- $m : M(s(y))$,
- $\alpha : \tau_1(y) = \mathsf{last}_{s(y)}(m)$,
- $p : m = \mathsf{rebuild}_{s(y)}(m)$.

Expanding the product $M(s(y)) \simeq M(y) \times A$, this is equivalent to:

- $m' : M(y)$,
- $z : A$,
- $\alpha : \tau_1(y) = \mathsf{last}_{s(y)}(\mathsf{pair}(m', z))$,
- $p : \mathsf{pair}(m', z) = \mathsf{rebuild}_{s(y)}(\mathsf{pair}(m', z))$.

This rewrites to:

- $m' : M(y)$,
- $z : A$,
- $\alpha : \tau_1(y) = \tau_1(z)$,
- $p : \mathsf{pair}(m', z) = \mathsf{pair}(\mathsf{rebuild}_y(m'), \mathsf{next}_y(m'))$.

Since τ_1 is an embedding, we may contract z with α:

- $m' : M(y)$,
- $p : \mathsf{pair}(m', y) = \mathsf{pair}(\mathsf{rebuild}_y(m'), \mathsf{next}_y(m'))$.

Splitting p into a pair of equalities, we recover $B(y)$. □

The strategy of the above proof is reminiscent of the proof of the rolling rule (Lemma 6.2). In particular, the definition of B is almost that of the fixpoints of an operation on $\sum_{x:A} M(x)$. However, type of p seems to resist this (it is an identification in $M(x)$, not $M(\mathsf{next}_x(m))$). It is unclear to us how this analogy can be exploited.

7.4 Defining the natural numbers

Lemma 7.8. *There is a stable natural number algebra that embeds into $\mathbb{Z}$.*

Proof. By Lemma 7.7, we have a stable natural number algebra A with a morphism $f : A \to \mathbb{Z}$. We now apply Construction 5.1 with $A \equiv_{\mathrm{def}} A$ and $B \equiv_{\mathrm{def}} \mathbf{1}$. The equivalence $A \simeq B \sqcup A$ is the structure map of the stable natural number algebra A. Note that B has a unique element. The resulting integer algebra (with carrier $A \sqcup \mathbf{1} \sqcup A$) again lies over $\mathbb{Z}$: we send $\tau_0(a)$ to $S^{-1}(\mathsf{inv}(a))$, $\tau_1(*)$ to Z, and $\tau_2(a)$ to $S(f(a))$. Here, inv is the underlying map of the unique integer algebra morphism from $\mathbb{Z}$ to $\mathbb{Z}'$ where $\mathbb{Z}'$ has automorphism S^{-1} instead of S.

By initiality of $\mathbb{Z}$, the algebra map $\mathbb{Z} \to A \sqcup \mathbf{1} \sqcup A$ is a section of the algebra map $A \sqcup \mathbf{1} \sqcup A \to \mathbb{Z}$. By construction, the former map sends $\mathbb{Z}^0$ in the decomposition (2) to the middle component in $A \sqcup \mathbf{1} \sqcup A$, and in turn, that middle component is sent to $\mathbb{Z}^0$ by the latter map (specifically, to Z). This exhibits Z^0 as a retract of $\mathbf{1}$, in particular it is contractible. We may thus silently replace Z^0 by $\mathbf{1}$ in the obtained decomposition (2).

This makes $\mathbf{1} \sqcup \mathbb{Z}^+$ with zero element $\tau_0(*)$ and successor function $\tau_1 \circ S^+$ into a stable natural number algebra embedding into $\mathbb{Z}$. ◻

In the last step of the above lemma, we can equivalently directly use $\mathbb{Z}^+$ as the desired natural number algebra. Denote the image of the zero in $\mathbb{Z}$ by t. We then need to postcompose with the shifting map of $\mathbb{Z}$ that sends t to Z to obtain the algebra embedding from $\mathbb{Z}^+$ to $\mathbb{Z}$.

Lemma 7.9. *Let A be a stable natural number algebra over $\mathbb{Z}$ such that $A(Z)$ is contractible. Then A is initial.*

Proof. Using Lemma 3.3, it suffices to show that A has elimination. In natural number algebras over A, having a section is a covariant structure. That is, given a morphism $E' \to E$ of natural number algebras over A, if E' has a section, then so does E. Using Lemma 7.7, it thus suffices to construct a section for a *stable* natural number algebra E over A. In fact, we will show that E is fiberwise contractible over A. Since both A and E are stable, we have $E(z) \simeq 1$ and $E(s(a)) \simeq E(a)$ for $x : \mathbb{Z}$ and $a : A(x)$.

We define an integer algebra Q over $\mathbb{Z}$ with underlying family of propositions

$$Q(x) \equiv_{\mathrm{def}} \prod_{a:A(x)} \mathsf{isContr}(E(a)).$$

Note that $Q(Z)$ holds because $z : 1 \to A(Z)$ is an equivalence and $E(z)$ is contractible. We will check in the next paragraph that $Q(x)$ and $Q(S(x))$ are logically equivalent for $x : \mathbb{Z}$. Once we have that, initiality of $\mathbb{Z}$ gives a section q of Q. Then we have $q(x, a) : \mathsf{isContr}(E(a))$ for $a : A(x)$ as desired.

From $s : A(x) \to A(S(x))$ and $E(s(a)) \simeq E(a)$, we have that $Q(S(x))$ implies $Q(x)$. Let us check that $Q(x)$ implies $Q(S(x))$. Given $f : \prod_{a:A(x)} \mathsf{isContr}(E(a))$ and $a' : A(S(x))$, we need to show that $E(a')$ is contractible. We case split on

$$[z, s]^{-1}(S(x), a') : \mathbf{1} \sqcup \textstyle\sum_{x:\mathbb{Z}} A(x).$$

- For $(S(x), a') = (Z, z)$, the goal follows from $E(z) \simeq 1$.
- For $(S(x), a') = (S(y), s(a))$ where $y : \mathbb{Z}$ and $a : A(y)$, it remains to show that $E(s(a))$ is contractible. This follows from $E(s(a)) \simeq E(a)$ and $f(a) : \mathsf{isContr}(E(a))$. ◻

Proof of Theorem 3.4. Apply Lemma 7.9 to Lemma 7.8. ◻

REFERENCES

[1] Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. 2023. Two-level type theory and applications. *Mathematical Structures in Computer Science* 33, 8 (2023), 688–743. https://doi.org/10.1017/S0960129523000130

[2] Ana Bove, Peter Dybjer, and Ulf Norell. 2009. A brief overview of Agda—a functional language with dependent types. In *Theorem Proving in Higher Order Logics*. Springer, 73–78.

[3] Martín Hötzel Escardó. 2019. Introduction to Univalent Foundations of Mathematics with Agda. Lecture notes from Midlands Graduate School. https://www.cs.bham.ac.uk/~mhe/HoTT-UF-in-Agda-Lecture-Notes/HoTT-UF-Agda.html#149931

[4] Jason J Hickey. 1996. Formal objects in type theory using very dependent types. *Foundations of Object Oriented Languages* 3 (1996), 117–170.

[5] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. 2018. The homotopy theory of type theories. *Advances in Mathematics* 337 (2018), 1–38. https://doi.org/10.1016/j.aim.2018.08.003

[6] Krzysztof Kapulkin and Karol Szumiło. 2019. Internal languages of finitely complete (∞, 1)-categories. *Selecta Mathematica* 25 (2019), 1–46.

[7] Nicolai Kraus and Jakob 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)*. IEEE, 1–13.

[8] Hoang Kim Nguyen and Taichi Uemura. 2022. ∞-type theories. arXiv:2205.00798 [math.CT]

[9] Nima Rasekh. 2021. Every Elementary Higher Topos has a Natural Number Object. *Theory and Applications of Categories* 37, 13 (2021), 337–377.

[10] Egbert Rijke. 2017. Discussion post on the nForum. https://nforum.ncatlab.org/discussion/6691/higher-inductive-type/?Focus=61552#Comment_61552

[11] Egbert Rijke. 2022. Introduction to Homotopy Type Theory. arXiv:2212.11082 [math.LO]

[12] Robert Rose. 2020. *The Natural Numbers in Predicative Univalent Type Theory.* Ph. D. Dissertation. Indiana University.

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