Functions out of Higher Truncations

Paolo Capriotti[1], Nicolai Kraus[*1], and Andrea Vezzosi[2]

1   University of Nottingham, United Kingdom
   `{ngk,pvc}@cs.nott.ac.uk`
2   Chalmers University of Technology, Sweden
   `vezzosi@chalmers.se`

──── **Abstract** ────

In homotopy type theory, the truncation operator $\|-\|_n$ (for a number $n \geq -1$) is often useful if one does not care about the higher structure of a type and wants to avoid coherence problems. However, its elimination principle only allows to eliminate into n-types, which makes it hard to construct functions $\|A\|_n \to B$ if B is not an n-type. This makes it desirable to derive more powerful elimination theorems. We show a first general result: If B is an $(n+1)$-type, then functions $\|A\|_n \to B$ correspond exactly to functions $A \to B$ which are constant on all $(n+1)$-st loop spaces. We give one "elementary" proof and one proof that uses a higher inductive type, both of which require some effort. As a sample application of our result, we show that we can construct "set-based" representations of 1-types, as long as they have "braided" loop spaces. The main result with one of its proofs and the application have been formalised in Agda.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases homotopy type theory, truncation elimination, constancy on loop spaces

Digital Object Identifier 10.4230/LIPIcs.CSL.2015.359

1   Introduction

As it is very well-known, the type constructor Σ of Martin-Löf type theory expresses a very strong form of existence. Although a type of the form $\Sigma\,(a : A)\,.\,P(a)$ is read as "there exists an element in A for which the predicate P holds" under the *propositions as types* view, an element of such a type is more than a proof of mere existence: it includes a very concrete example of an element $a : A$. This is not always satisfying as, for example, the set-theoretic axiom of choice becomes a tautology when translated naively to type theory. The idea of adding a construction which allows to formulate existence in a weaker sense has been studied intensively in various different settings. As far as we know, the first documented appearance are *squash types* in the extensional theory of NuPRL [7]. Later, Awodey and Bauer introduced a similar concept in extensional Martin-Löf type theory, called *bracket types* [4]. Homotopy type theory has introduced the *propositional truncation* operation, written $\|-\|_{-1}$ or simply $\|-\|$ [14]. It forces all elements to be equal, in the sense that the identity type $x = y$ is inhabited for any $x, y : \|A\|_{-1}$, and it is well-known that $x = y$ will in fact be uniquely inhabited (i.e. equivalent, or isomorphic, to the unit type). Classically, $\|A\|_{-1}$ is always equivalent to either the unit type or the empty type, but this is of course not the case in a constructive setting.

─────────────

The homotopical view has suggested that propositional truncation is only one out of infinitely many operations that reduce the complexity of a type. As "types are weak ω-groupoids" ([12] and [15]), it is easy to imagine that there is, for every number $n \geq -1$, an operation which trivialises all the structure above level $(n + 1)$. In other words, this is a reflector for the category of weak n-groupoids, viewed as a subcategory of weak ω-groupoids, roughly speaking. In homotopy type theory, we write this operation as $\|-\|_n$ ("n-truncation"), and it can be seen and implemented as a *higher inductive type* [14]. The truncation operator $\|-\|_n$ is a *monad* in some appropriate sense (and even a *modality* in the sense of [14]), and if we want to, we can choose to work completely in that monad. Types that are canonically equivalent to their n-truncation are called *n-types*, or *n-truncated types*.

Considering n-types (for some given n) instead of all types is useful if we do not care about or want to avoid potential higher equality proofs. For example, if we formalise algebraic structures such as groups, we may require that the type of group elements is of truncation level 0 in order to match the set-theoretic definition: equality of group elements should be a mere proposition and not carry additional information, that is, there is at most one proof that given group elements are equal. As a consequence, for any type A with an element $a : A$, the type $a = a$ is not necessarily a group. It does have a neutral element an elements can be inverted and composed, corresponding to the fact that equality is reflexive, symmetric, and transitive. However, $a = a$ is not a 0-truncated type. We can use 0-truncation to make up for this, and $\|a = a\|_0$ is indeed a group, called the *fundamental group* of A at basepoint a, while $a = a$ (as *pointed type* also written $\Omega(A, a)$) is the *loop space* at point a.

A drawback of truncations is that it can be hard to get out of them, that is, "to leave the monad". A priori we have, for any type A and number $n \geq -1$, a map $|-| : A \to \|A\|_n$, but there is in general no function in the other direction. The universal property of $\|-\|_n$ says that, via composition with $|-|$, the type of functions $\|A\|_n \to B$ is equivalent to the type $A \to B$, but only if B is n-truncated. To continue with the previous example, an element of the fundamental group of A at basepoint a is really an equivalence class of equality proofs (or *paths*) between a and itself, but it is in general impossible to get a specific representative from such a class; that is, we cannot construct a section of the map $|-| : (a = a) \to \|a = a\|_0$. Of course, we would not have expected anything else: it is unreasonable to assume that we can make this sort of choice without any further assumptions. Although the truncation operator $\|-\|_n$ is often described as "cutting of" higher structure of a type, it is more accurate to think of it as "filling non-trivial loops", which makes it plausible that it is *harder* instead of *easier* to define a function out of $\|A\|_n$ than out of A.

Unlike in the example above, it is in some cases reasonable to expect that we can get a function $\|A\|_n \to B$ even if B is not an n-type. If $\|A\|_{-1}$ tells us that A has some element without revealing a concrete one to us, then a function $\|A\|_{-1} \to B$ should be the same as a function $f : A \to B$ which cannot look at the "input".[1] What exactly this means is difficult to state in general (see [8]), so let us restrict ourselves to the case that B is 0-truncated (also called a *set*). In this case, the statement that "f does not look at its input" can be expressed by saying that f maps any pair of inputs to equal values, $\Pi_{x,y:A}(f(x) = f(y))$. Indeed, it has been shown that a function f with this behaviour gives rise to a map $\|A\|_{-1} \to B$ [10].

Even if we have a function $A \to B$, it can be very hard to tell whether it is possible to construct a function $\|A\|_n \to B$ unless B is an n-type, and if it is possible, there is no direct

---

[1] This only makes sense if stated internally. Of course, a concrete implementation of f *can* compute differently if applied to different terms of type A. As long as we stay inside the theory, we cannot talk about judgmental equality.

way to do so as the universal property (or the elimination principle) cannot be applied directly. The usual workaround is looking for an n-type C "in the middle", that is such that one has functions $A \to C$ and $C \to B$. One can then apply the elimination principle to construct a function $\|A\|_n \to C$ which, by composition, yields a function $\|A\|_n \to B$ as desired. The type C is constructed ad-hoc, and it is natural to ask for a more powerful elimination principle (or universal property) of $\|-\|_n$ which allows the construction of functions $\|A\|_n \to B$ in a more principled and streamlined way.

This has been done for the (-1)-truncation in previous work [8], where it is shown that functions $\|A\|_{-1} \to B$ correspond exactly to functions $A \to B$ with an infinite tower of coherence conditions. This can be understood as a generalised version of the usual universal property of $\|-\|_{-1}$. If B is known to be n-truncated for some fixed finite n, the infinite tower becomes finite and can be expressed directly in type theory, whereas the existence of *Reedy limits* [13] is necessary for the general case. If B is a 0-type, the "tower" of coherence condition is exactly the single condition $\Pi_{x,y:A}(f(x) = f(y))$ discussed above. If B is even a (-1)-type itself, the tower vanishes completely and the usual universal property remains. Unfortunately, it seems that there is no immediate generalisation of the proof of [8] to n-truncations.

In this paper, we do consider n-truncations for general n, but we assume that B is $(n + 1)$-truncated, and already this case seems to be involved. We show that functions $\|A\|_n \to B$ correspond exactly to those functions $A \to B$ that are constant on all $(n + 1)$-st loop spaces. We offer two proofs for this fact, one which works in "plain" homotopy type theory with general truncations, and the other involving a higher inductive type. The first proof, which we call the "elementary proof", is close to not even requiring the univalence axiom (the central concept of homotopy type theory expressing that equality in the universe is given by type equivalence). The only reason why univalence is necessary is that we need to be able to translate between truncations ($\|a =_A b\|_n$ is equivalent to $|a| =_{\|A\|_{n+1}} |b|$). The second proof (Section 4) uses an argument that makes crucial use of both a higher inductive type and the univalence axiom, and we therefore call it the "HIT proof". In the HIT proof, we will construct a higher inductive type in such a way that it is the "initial" type through which functions $f : A \to B$ with the property (10) factor, and we will show that this type *is* really $\|A\|_n$. Although we show an equivalence of types, we believe that the main application is the construction of functions $\|A\|_n \to B$, that is, one may often want to use only one direction of the equivalence. Therefore, the result can be used as an elimination principle that is more powerful than the usual recursion principle of the truncation. We also present a sample application (a translation of types into "set-based representation"), and conclude with a discussion on how the generalised statement should look like, and under which assumptions it should be provable.

The main contents of this paper have, in slightly different form, appeared in the second-named author's Ph.D. thesis [9].

Outline. We start by stating the result of the paper in Section 2, and discuss two special cases ($n \equiv -1$ and $n \equiv 0$). In Section 3, we give the "elementary" proof of this result, and in Section 4, the (technically harder, but conceptually clear) proof that uses a higher inductive type. We discuss a sample application of the case $n \equiv 0$ in Section 5, namely a construction of a *set-based representation* of any given type, provided that it fulfils a property that e.g. loop spaces do. Finally, in Section 6, we compare the two proofs with each other. We also compare our result with the *general universal property of the propositional truncation* as proved before [8], and discuss why the potential generalisations seem so much more involved than what we have done here.

Setting.   We consider the theory of the standard reference on homotopy type theory, that is, the textbook [14]. To summarise, we need a version of intensional Martin-Löf type theory with Σ, Π, and identity types. In addition, we assume that the theory has a univalent universe, and that there are truncation operators $\|-\|_n$ for all $n \geq -1$, with the canonical projections $|-| : A \to \|A\|_n$. This concept is explained in detail in [14, Chap. 7.3]). The statement and the first proof that we give do not need higher inductive types [14, Chap. 6] other than the truncations, while the second proof that we give makes heavy use of such a higher inductive type.

Agda Formalisation.   We have formalised the main result, together with the "elementary" proof (Section 3) and the sample application (Section 5), in Agda [6]. The source code can be found on GitHub, at `github.com/pcapriotti/agda-base/tree/trunc`. The results of this paper are contained in the module `hott.truncation.elim`. A browsable HTML version of the formalisation can be accessed at `paolocapriotti.com/agda-base/trunc/` `hott/truncation/elim.html`. We encourage a reader who is not familiar with Agda to have a look at the latter, which does not need any software apart from a web browser. For all the technical details, we refer to the readme file in the repository.

On a minor note, we have chosen not to make use of the common (but, as far as we know, not justified by a formal argument) hack that makes truncations satisfy the judgmental computation rule. As we wanted our formalisation to be readable, this has required us to think of some implementation strategies that make the code in this setting more elegant than the "straightforward" formalisation approaches.

2   The Statement of the Theorem

Let us begin by clarifying some notation. In general, we stick closely to the terminology of the standard reference on the topic, the textbook [14]. We write $\Pi_{a:A}B(a)$ for Π-types as it is done there, but $\Sigma\,(a : A)\,.\,B(a)$ for Σ-types.[2] For better readability, we uncurry implicitly and write $f(a, b) : C$, even if f is a function of type $A \to B \to C$. Instead of $\lambda h.h \circ g$, we write $\_\circ g$. By the *distributivity law of Σ and Π*, we mean the well-known equivalence

$$\Pi_{a:A}\Sigma\,(b : B(a))\,.\,C(a, b) \;\simeq\; \Sigma\,(g : \Pi_{a:A}B(a))\,.\,\Pi_{a:A}C(a, g(a)), \tag{1}$$

sometimes called the *type-theoretic axiom of choice*. As it is standard [14], we write is-n-type(A) for the propositional type expressing that A is n-truncated if $n \geq -2$ is an integer, defined by

$$\text{is-}(-2)\text{-type}(A) :\equiv \Sigma\,(a_0 : A)\,.\,\Pi_{a:A}a = a_0 \tag{2}$$

$$\text{is-}(n+1)\text{-type}(A) :\equiv \Pi_{a_1, a_2:A}\text{is-}n\text{-type}(a_1 = a_2), \tag{3}$$

and the special case when n is -2 ("A is contractible") is also written as isContr(A). We assume that there is a universe $\mathcal{U}$, and we write $\mathcal{U}^n$ for the type (or "universe") of n-types in $\mathcal{U}$ (cf. [14, Chap. 7.1]),

$$\mathcal{U}^n :\equiv \Sigma\,(X : \mathcal{U})\,.\,\text{is-}n\text{-type}(X). \tag{4}$$

---

2   This seemingly inconsistent notation is intentional: we sometimes have nested Σ-types, e.g. $\Sigma\,(a : A)\,.\,\Sigma\,(b : B(a))\,.\,C(a, b)$, and we view the components as "equally valued"; thus, writing exactly one component bigger than the others would not look correct.

Further, we write $\mathcal{U}_\bullet$ for the type (or "universe") of pointed types in $\mathcal{U}$ (cf. [14, Def. 2.1.7]),

$$\mathcal{U}_\bullet :\equiv \Sigma\,(X : \mathcal{U})\,.\,X. \tag{5}$$

If we have a type A and a pointed type (B, b), together with a function $f : A \to B$, we say that "f is null" if it is constantly b, that is,

$$\mathsf{isNull}(f) :\equiv \Pi_{x:A} b = f(x). \tag{6}$$

Recall that there is an endofunction on $\mathcal{U}_\bullet$, the *loop space function* Ω,

$$\Omega(A, a) :\equiv (a = a, \mathsf{refl}_a). \tag{7}$$

For any natural number n, we can iterate this endofunction n times, for which we write Ω^n. Instead of $\pi_1\,(\Omega^n(A, a))$ and $\pi_1((\Omega(A, a)))$, we simply write $\Omega^n_t(A, a)$ and $\Omega_t(A, a)$ if we want to talk about the underlying type (i.e. ignore the point). Further, given two types A and B together with any function $f : A \to B$ and a point $a : A$, we have a function

$$\mathsf{ap}_{f,a} : \Omega_t(A, a) \to \Omega_t(B, f(a)). \tag{8}$$

In the same way, we have (given A, B, f as before) $\mathsf{ap}^n_{f,a} : \Omega^n_t(A, a) \to \Omega^n_t(B, f(a))$, and Ω is really an endofunctor in some appropriate sense.[3]

Our result can now be stated as follows:

▶ **Theorem 1.** *Let $n \geq -1$ be a number, A a type, and B an $(n + 1)$-type. Assume that $f : A \to B$ is a function. Then, f can be factored through the n-truncation, that is*

$$\Sigma\,(f' : \|A\|_n \to B)\,.\,f' \circ |-| = f, \tag{9}$$

if and only if $\mathsf{ap}^{n+1}_{f,a}$ is null for every a,

$$\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}^{n+1}_{f,a}), \tag{10}$$

and both of the types (9) *and* (10) *are propositional.*

An immediate corollary tells us how we can eliminate out of truncations:

▶ **Corollary 2.** *Assume we have n, A and B as in Theorem 1. If we want to construct a function $\|A\|_n \to B$, it suffices to find a function $f : A \to B$ which satisfies $\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}^{n+1}_{f,a})$.*

Before approaching a proof of Theorem 1, let us have a look at two special cases, namely the cases $n \equiv -1$ and $n \equiv 0$. The first case is known [10] and will serve as the base case for the two general proofs presented later. The second case is not strictly necessary, but serves to exemplify the techniques used in the "HIT proof" (Section 4).

The case $n \equiv -1$. The simplified statement of Theorem 1 reads in this case as follows: Assume we are given a type A and a 0-type B (often called a *set*). A function $f : A \to B$ factors through the propositional truncation if and only if

$$\Pi_{x,y:A} f(x) = f(y). \tag{11}$$

This follows easily from previous work, e.g. [8, Prop. 2.2]. It is a pleasant surprise that "$\mathsf{ap}^0_{f,a}$ is null for all a", simply by unfolding our definitions, simplifies to (11), which is "f is constant" in the sense of [10].[4]

---

[3] Of course, $\mathsf{ap}_{f,a}$ is its action on the morphism f and could thus rightfully be called $\Omega(f, a)$.

[4] In the simplified formulation, we have omitted the part that the two logically equivalent types are propositional. This is easy to see here, and will in the general case be part of the proof.

$$\|A\|_n \to B \xrightarrow{\quad \mathfrak{c}_n \quad} \Sigma\,(f : A \to B)\,.\,\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1})$$

$$\underset{\_\,\circ\,|-|}{\searrow} \quad A \to B \quad \underset{\pi_1}{\swarrow}$$

Figure 1 The canonical map $\mathfrak{c}_n$ as map between fibres.

The case $n \equiv 0$.   Here, our result (Theorem 1) implies that, for any type A and 1-type B, a function $f : A \to B$ factors through $\|A\|_0$ if and only if, for all $a : A$ and $p : a = a$, we have that $\mathsf{ap}_{f,a}(p)$ equals $\mathsf{refl}_{f(a)}$. As Shulman has remarked in an online discussion (in the comment section of a blog post [5]), this follows from the *Rezk completion* [1]: Let $\tilde{A}$ be the precategory with the type A of objects and $\mathsf{hom}(a_1, a_2) :\equiv \|a_1 =_A a_2\|_{-1}$, and let $\tilde{B}$ be the category with B as objects and $\mathsf{hom}(b_1, b_2) :\equiv (b_1 =_B b_2)$. Then, f with the condition $\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{f,a})$ gives (already using the case $n \equiv -1$) rise to a functor $\tilde{A} \to \tilde{B}$. Such a functor generates a functor between the Rezk completion of $\tilde{A}$ and the category $\tilde{B}$, and the former happens to be $\|A\|_0$.

In the remainder of the current section, we give a simple technical construction which essentially serves as a reformulation of Theorem 1 and which is necessary for both the elementary and the HIT proof. For types A and B, assume we are given a function $g : \|A\|_n \to B$. We can consider the composition $A \xrightarrow{|-|} \|A\|_n \xrightarrow{g} B$. For any $a : A$ we have, by functoriality of Ω^{n+1}, that the composition

$$\Omega_t^{n+1}(A, a) \xrightarrow{\mathsf{ap}_{|-|,a}^{n+1}} \Omega_t^{n+1}(\|A\|_n, |a|) \xrightarrow{\mathsf{ap}_{g,|a|}^{n+1}} \Omega_t^{n+1}(B, g(|a|)) \tag{12}$$

is equal to $\mathsf{ap}_{g\circ|-|,a}^{n+1}$. But $\Omega_t^{n+1}(\|A\|_n, |a|)$ is contractible ([14, Thm. 7.2.9]), and $\mathsf{ap}_{g,|a|}^{n+1}$ clearly maps its unique element to the basepoint of $\Omega^{n+1}(B, g(|a|))$. Therefore, $\mathsf{ap}_{g\circ|-|,a}^{n+1}$ is null. From this construction, we get a canonical function

$$\mathfrak{c}_n : (\|A\|_n \to B) \to \Sigma\,(f : A \to B)\,.\,\left(\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1})\right). \tag{13}$$

We then claim the following:

▶ **Lemma 3** ("Total space" formulation of Theorem 1). *For any $n \geq -1$, any type A and any $(n+1)$-type B, the types $\|A\|_n \to B$ and $\Sigma\,(f : A \to B)\,.\,\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1})$ are equivalent, and the equivalence is given by the canonical function $\mathfrak{c}_n$.*

It is easy to see that Lemma 3 does indeed imply, and is nearly immediately equivalent to, Theorem 1. Consider the triangle shown in Figure 1, where the top horizontal map is the canonical map $\mathfrak{c}_n$, the left one is composition with $|-|$, and the right one is simply the projection. The triangle clearly commutes (judgmentally) by construction. Let us fix some function $f : A \to B$. The fibre (or "inverse image") over f is, in the case of $\_ \circ |-|$, exactly (9), i.e. the statement that f can be lifted. In the second case, the fibre is (10). Therefore, $\mathfrak{c}_n$ induces an equivalence of the two fibres, which implies that $\mathfrak{c}_n$ itself is an equivalence (see [14, Thm. 4.7.7]).

3   The "Elementary" Proof

In this section, we give our first proof of Lemma 3 (and thereby of Theorem 1). This does not need higher inductive types apart from truncations that already appear in the statement.

The idea is to not prove the result for *any* type A first, but only for an n-connected one.[5] Afterwards, we generalise this to arbitrary types, by splitting the type into its "connected components" and gluing together the constructions for the components.

▶ **Lemma 4.** *If $n \geq -1$ be a number, A an n-connected type, and B be an $(n+1)$-type, the canonical map $\mathfrak{c}_n$ is an equivalence.*

Proof. We do induction on n. As already discussed above, the case that n is -1 is known (e.g. [8, Prop. 2.2]).

Let now $n \geq 0$ be any given number. Note that, due to the assumption that $\|A\|_n$ is contractible, we have a unique element $x_0 : \|A\|_n$, the type $\|A\|_n \to B$ is actually equivalent to B, and any function $g : \|A\|_n \to B$ is uniquely specified by its value $g(x_0)$.

The claim of the lemma is propositional. Applying the eliminator of $\|A\|_n$, we may not only assume that we are given $x_0 : \|A\|_n$, but we can also assume a point $a : A$. A potential inverse of $\mathfrak{c}_n$ is then given by[6]

$$\mathfrak{d}_n : \left(\Sigma \, (f : A \to B) \, . \, \Pi_{a:A} \mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1}) \right) \to (\|A\|_n \to B) \tag{14}$$

$$\mathfrak{d}_n(f, p) :\equiv \lambda \_ . f(a). \tag{15}$$

To show that $\mathfrak{c}_n$ and $\mathfrak{d}_n$ are inverses, we check that both compositions are the identities. One direction is easy: for any $g : \|A\|_n \to B$, we have

$$\mathfrak{d}_n(\mathfrak{c}_n(g))(x_0) \equiv g(|a|), \tag{16}$$

and the latter is equal to $g(x_0)$.

For the other direction, assume we have $f : A \to B$ together with a proof q. We need to show $(f, q) = \mathfrak{c}_n(\mathfrak{d}_n(f, q))$. Fortunately, the equality of the two second components is automatic thanks to the fact that $\mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1})$ is propositional, and we only need to prove the equality of f and $\pi_1(\mathfrak{c}_n(\mathfrak{d}_n(f, q)))$. We observe that the latter expression computes to $\lambda \_ . f(a)$. Thus, our goal is to show that, for any $a' : A$, we have $f(a) = f(a')$.

We use the induction hypothesis with $(a = a')$ for A, and $f(a) = f(a')$ for B. By the connectedness assumption on A, the type $|a| = |a'|$ is contractible. Consequently, the type $\|a = a'\|_{n-1}$ is contractible ([14, Thm. 7.3.12], note that this theorem depends on the univalence axiom). Put differently, $(a = a')$ is $(n-1)$-connected. As B is an $(n+1)$-type, we know that $f(a) = f(a')$ is n-truncated. By the induction hypothesis, it is hence enough to construct an element of

$$\Sigma \, (k : a = a' \to f(a) = f(a')) \, . \, \Pi_{p:a=a'} \mathsf{isNull}(\mathsf{ap}_{k,p}^n). \tag{17}$$

For k, we choose ap_f. By path induction, we may assume that p is refl_a. Thus, we need to show that $\mathsf{ap}_{\mathsf{ap}_{f,a},\mathsf{refl}_a}^n$ is null. This term is equal to $\mathsf{ap}_{f,a}^{n+1}$.[7] The condition that this function null is exactly what is given by $q(a')$. ◀

To move from n-connected to arbitrary types A, we simply split a type into n-connected components. This is very intuitive for $n \equiv 0$, in which case we use that any type (or "space") can be viewed as the "disjoint sum" of its connected components. To be precise, an element

---

[5] Recall that a type A is n-connected if $\|A\|_n$ is contractible [14, Def. 7.5.1].

[6] We use $\_$ if we do not need to give the bound variable a name.

[7] Depending on the the exact definition of ap^n, this can hold judgmentally, but can also be rather involved. We refer to our formalisation for technical details.

of a component is a point of A together with a proof that it is in the component. For $n \equiv 0$, this proof is propositional. For higher n, it is not. This makes the general case less intuitive and hard to picture. In fact, the proof determines in which component the element is, which makes it seem circular. Fortunately, it is easier to write down the type-theoretic argument than picturing the topological intuition, as we will see in the following lemma.

▶ **Lemma 5.** *For any type A and number n, we define the family of n-connected components,*

$$\mathsf{conn}_n : \|A\|_n \to \mathcal{U} \tag{18}$$

$$\mathsf{conn}_n(x) :\equiv \Sigma\, (a : A)\, .\, x =_{\|A\|_n} |a|. \tag{19}$$

Then, for any $x : \|A\|_n$, the type $\mathsf{conn}_n(x)$ is n-connected. Further, "choosing an n-connected component and then a point in this component" corresponds to "choosing a point", that is,

$$\Sigma\, (x : \|A\|_n)\, .\, \mathsf{conn}_n(x) \;\simeq\; A. \tag{20}$$

Proof. This is easy and standard. For the first part, we claim that the equivalence

$$\left\| \Sigma\, (a : A)\, .\, x =_{\|A\|_n} |a| \right\|_n \;\simeq\; \Sigma\, (y : \|A\|_n)\, .\, x =_{\|A\|_n} y \tag{21}$$

holds, where the left-hand type is $\|\mathsf{conn}_n(x)\|_n$ by definition, and the right-hand type has the form of a *singleton*.[8] For both directions of (21), we apply the dependent eliminator of $\|-\|_n$. From left to right, we map $|(a,p)|$ to $(|a|,p)$. From right to left, we map $(|a|,p)$ to $(|a|,p)$. For an alternative proof, see [14, Cor. 7.5.8].

To see that the equivalence (20) holds, it is enough to unfold the definition of conn_n, and use that in $\Sigma\, (x : \|A\|_n)\, .\, \Sigma\, (a : A)\, .\, x =_{\|A\|_n} |a|$, the first and the third component form a singleton. ◀

Finally, we can complete the first proof of our main result:

"Elementary" proof of Lemma 3. Assume we have n, A, and B as in the statement. The preceding two lemmata tell us that, for any $x : \|A\|_n$, the canonical map

$$\mathfrak{c}_n^x : B \to \left(\Sigma\, (f_x : \mathsf{conn}_n(x) \to B)\, .\, \Pi_{y:\mathsf{conn}_n(x)} \mathsf{isNull}(\mathsf{ap}_{f_x,y}^{n+1}) \right) \tag{22}$$

is an equivalence (note that we have omitted the contractible type $\|\mathsf{conn}_n(x)\|_n$ in the domain of $\mathfrak{c}_n^x$). A family of equivalences gives rise to an equivalence of families, so that we get that the map

$$\tilde{\mathfrak{c}}_n : (\|A\|_n \to B) \;\to\; \left(\Pi_{x:\|A\|_n} \Sigma\, (g_x : \mathsf{conn}_n(x) \to B)\, .\, \Pi_{y:\mathsf{conn}_n(x)} \mathsf{isNull}(\mathsf{ap}_{g,y}^{n+1}) \right) \tag{23}$$

$$\tilde{\mathfrak{c}}_n(k) :\equiv \lambda x. \mathfrak{c}_n^x(k(x)) \tag{24}$$

is also an equivalence.

All we need at this point is an equivalence from the codomain of the function (24) to the type stated in the theorem, i.e. $\Sigma\, (f : A \to B)\, .\, \Pi_{a:A} \mathsf{isNull}(\mathsf{ap}_{f,a}^{n+1})$, and the composition of (24) and this equivalence has to be the canonical map $\mathfrak{c}_n$. We calculate:

$$\Pi_{x:\|A\|_n} \Sigma\, (g_x : \mathsf{conn}_n(x) \to B)\, .\, \Pi_{y:\mathsf{conn}_n(x)} \mathsf{isNull}(\mathsf{ap}_{g_x,y}^{n+1}) \tag{25}$$

---

[8] If $z_0 : Z$ is some point of some type, we call any type of the form $\Sigma\, (z : Z)\, .\, z = z_0$ a *singleton*. It is well-known that singletons are contractible and therefore "neutral" components of Σ-types, which we use here and later.

(by the distributivity law)

$$\simeq \quad \Sigma\left(g : \Pi_{x:\|A\|_n}(\mathsf{conn}_n(x) \to B)\right).\Pi_{x:\|A\|_n}\Pi_{y:\mathsf{conn}_n(x)}\mathsf{isNull}(\mathsf{ap}_{g(x),y}^{n+1}) \tag{26}$$

(by currying and using the canonical equivalence (20))

$$\simeq \quad \Sigma\left(h : A \to B\right).\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{\lambda y:\mathsf{conn}_n(|a|).h(\pi_1 y),(a,\mathsf{refl}_{|a|})}^{n+1}) \tag{27}$$

Fortunately, the (pointed) types $\Omega^{n+1}(\mathsf{conn}_n(|a|),(a,\mathsf{refl}_{|a|}))$ and $\Omega^{n+1}(A,a)$ are equivalent, with the equivalence being $\mathsf{ap}_{\pi_1}^{n+1}$; this is an easy technical statement that follows from [11, Lem. 5.1]. If we compose $\mathsf{ap}_{\lambda y:\mathsf{conn}_n(|a|).h(\pi_1 y),(a,\mathsf{refl}_{|a|})}^{n+1}$ with the inverse of this equivalence, functoriality of ap^{n+1} allows us to simplify the expression.

$$\simeq \quad \Sigma\left(h : A \to B\right).\Pi_{a:A}\mathsf{isNull}(\mathsf{ap}_{h,a}^{n+1}) \tag{28}$$

We need to check that the composition of $\tilde{\mathfrak{c}}_n$ with this equivalence is indeed the canonical function $\mathfrak{c}_n$. This is immediate as we only need to check that the first component (the map $A \to B$) turns out to be the correct function, as the second component is propositional. ◀

4   The "HIT Proof"

Our second proof is fairly technical. We construct a higher inductive type with a suitable elimination property and show that it is equivalent to $\|A\|_n$. As a preparation, we show a small lemma. It is a part of a theorem that has been introduced in [9], where it is described as *local generalised Hedberg argument*.

▶ **Lemma 6** (main part of [9, Thm. 3.2.1]). *Let (A, a_0) be a pointed type. Assume further that P is a* pointed *family of $(n-1)$-types over (A, a_0), that is, a family $P : A \to \mathcal{U}^{n-1}$ with a point $p_0 : P(a_0)$. If $P(a)$ implies that a_0 is equal to a, i.e. $m : \Pi_{a:A}P(a) \to a_0 = a$, then A is "locally an n-type" in the sense that $\Omega^{n+1}(A, a_0)$ is contractible.*[9]

Proof sketch. Consider the following composition of three maps, for any $a : A$:

$$a_0 = a \xrightarrow{\quad q \mapsto \mathsf{transport}^P(q,p_0) \quad} P(a) \xrightarrow{\quad m_a \quad} a_0 = a \xrightarrow{\quad q \mapsto m_{a_0}(p_0) \cdot q \quad} a_0 = a$$

By path induction, we easily see that these maps make $a_0 = a$ a retract of $P(a)$. Hence, the former is $(n-1)$-truncated [14, Thm. 7.1.4], which shows the claim [14, Thm. 7.2.9]. ◀

We are ready to define the higher inductive type that plays the central role in the second proof of Lemma 3. For the following definition and for the rest of the section, we fix a type A and a number $n \geq -1$.

▶ **Definition 7.** Define the higher inductive type H, which depends on A and n, as given by the constructors

$$\eta : A \to H \tag{29}$$

$$\epsilon : \Pi_{a,b:A}\left(\|a = b\|_{n-1} \to \eta(a) = \eta(b)\right) \tag{30}$$

$$\delta : \Pi_{a:A}\left(\mathsf{refl}_{\eta(a)} =_{\eta(a)=\eta(a)} \epsilon(a,a,|\mathsf{refl}_a|)\right) \tag{31}$$

$$t : \mathsf{is}\text{-}(n+1)\text{-}\mathsf{type}(H). \tag{32}$$

---

[9] This "local" form directly implies the "global" form: We can consider a relation $R : A \times A \to \mathcal{U}^{n-1}$ which implies identity and which has points $r_a : R(a,a)$ for all $a : A$; then, the lemma shows that A is an n-type.

The complicated looking constructors ϵ and δ are more intuitive than they looks at first sight. If we have $(a = b)$, we of course always get a proof of $\eta(a) = \eta(b)$ using ap_η. The constructor ϵ says that $\|a = b\|_{n-1}$ is sufficient, while δ ensures that ϵ is really a lifting of ap_η through $\|a = b\|_{n-1}$. This is because we could have used the expanded form

$$\delta' : \Pi_{a,b:A}\Pi_{p:a=b} \left(\mathsf{ap}_\eta(p) =_{\eta(a)=\eta(b)} \epsilon(a, b, |p|) \right), \tag{33}$$

instead of the constructor δ. By path induction on p, the type (33) is easily seen to be equivalent to the original type (31). While (33) might look more regular next to (30), we choose (31) just for simplicity.

The recursion principle for H is straightforward to write down. Given some $(n+1)$-type B, we need a function $f : A \to B$, together with a function $k : \Pi_{a,b:A}(\|a = b\|_{n-1}) \to f(a) = f(b)$ and a proof $h : \Pi_{a:A}\mathsf{refl}_{f(a)} =_{f(a)=f(a)} k(a, a, |\mathsf{refl}_{f(a)}|)$, we get a function $H \to B$ with the expected properties. It is more involved, nevertheless not inherently difficult, to state the induction principle following the standard ("intuitive") approach as used in [14, Chap. 6]. Given an $(n + 1)$-truncated family $P : H \to \mathcal{U}^{n+1}$, in order to prove $\Pi_{x:H}P(x)$, we need

$$\overline{\eta} : \Pi_{a:A}P(\eta(a)) \tag{34}$$

$$\overline{\epsilon} : \Pi_{a,b:A}\Pi_{q:\|a=b\|_{n-1}}\mathsf{transport}^P\left(\epsilon(a, b, q), \overline{\eta}(a)\right) =_{P(\eta(b))} \overline{\eta}(b) \tag{35}$$

$$\overline{\delta} : \Pi_{a:A}\left(\mathsf{transport}^{\lambda r.\mathsf{transport}^P(r,\overline{\eta}(a))=\overline{\eta}(a)}\left(\delta(a), \mathsf{refl}_{\overline{\eta}(a)}\right) = \overline{\epsilon}(a, a, |\mathsf{refl}_a|)\right). \tag{36}$$

The above type expressions look rather involved. Fortunately, we do not need to deal too much with them at all because we are only interested in the case that P is n-truncated (instead of, more generally, $(n + 1)$-truncated), which enables us to use the following observation:

▶ **Lemma 8** (Restricted dep. universal property of H). *Given A and $n \geq -1$ as above and a family of n-types, $P : H \to \mathcal{U}^n$, the canonical map*

$$\Pi_{x:H}P(x) \xrightarrow{\ -\circ\eta\ } \Pi_{a:A}P(\eta(a)) \tag{37}$$

is an equivalence.

Proof. As P is a family of n-types, the type $\mathsf{transport}^P\left(\epsilon(a, b, q), \overline{\eta}(a)\right) =_{P(\eta(b))} \overline{\eta}(b)$, appearing in (35) as the target of $\overline{\epsilon}$, is $(n - 1)$-truncated. By the standard universal property of the $(n - 1)$-truncation, we may thus assume that the q in the type (35) is of the form $|p|$ with $p : a = b$, and then do path induction on p. This shows that the type of $\overline{\epsilon}$ is equivalent to

$$\overline{\epsilon}'' : \Pi_{a:A}\mathsf{transport}^P\left(\epsilon(a, a, |\mathsf{refl}_a|), \overline{\eta}(a)\right) =_{P(\eta(a))} \overline{\eta}(a). \tag{38}$$

Under this equivalence, the type of $\overline{\delta}$ becomes

$$\overline{\delta}'' : \Pi_{a:A}\left(\mathsf{transport}^{\lambda r.\mathsf{transport}^P(r,\overline{\eta}(a))=\overline{\eta}(a)}\left(\delta(a), \mathsf{refl}_{\overline{\eta}(a)}\right) = \overline{\epsilon}''(a)\right). \tag{39}$$

We see that the dependent pair of (38) and (39) forms a family of singletons. Therefore, there is always a canonical and unique choice for $\overline{\epsilon}$ and $\overline{\delta}$. The induction principle can therefore be simplified to only (34). Let us write $\mathsf{rind} : \Pi_{a:A}P(\eta(a)) \to \Pi_{x:H}P(x)$ for this *restricted induction principle*. It is easy to check that rind is indeed an inverse of the map $\_ \circ \eta$:

- For any $f : \Pi_{a:A}P(\eta(a))$ and $a : A$, the expression $(\mathsf{rind}(f) \circ \eta)(a)$ can be reduced to $f(a)$.
- For any $g : \Pi_{x:H}P(x)$, assume $x : H$. We need to show $(\mathsf{rind}(g \circ \eta))(x) = g(x)$. Using the restricted induction principle, we may assume $x \equiv \eta(a)$, and the left side can be reduced to the right side of the equation. ◀

This allows us to conclude the following crucial property of H:

▶ **Lemma 9.** *The type H is n-truncated.*

Proof. It suffices to show that $\Omega^{n+1}(H, x)$ is contractible for all $x : H$ [14, Lem. 7.2.9]. The restricted induction principle of H tells us that, in order to show $P(x) :\equiv \mathsf{isContr}\left(\Omega^{n+1}(H, x)\right)$ for all x, we only need to prove $P(\eta(a_0))$ for any $a_0 : A$. Let us define a type family $Q : H \to \mathcal{U}^{n-1}$ using the restricted induction principle, $Q(\eta(a)) :\equiv \|a_0 = a\|_{n-1}$. This family is trivially inhabited at a_0. We want to show that Q implies local equality in the sense of $\Pi_{x:H}\left(Q(x) \to \eta(a_0) = x\right)$, and as this type family is n-truncated, we apply the restricted induction principle again and the goal becomes

$$\Pi_{a:A}\left(Q(\eta(a)) \to \eta(a_0) = \eta(a)\right). \tag{40}$$

By definition of Q, this is exactly given by the constructor ϵ, applied on a_0 and a.

This allows us to conclude, by Lemma 6, that H is n-truncated, as claimed.   ◀

It is straightforward and standard that an n-truncated type which satisfies the dependent eliminating principle of $\|A\|_n$ is necessarily equivalent to $\|A\|_n$, and we record:

▶ **Corollary 10.** *The types H and $\|A\|_n$ are equivalent.*

At the same time, we have the following:

▶ **Lemma 11** (Universal property of H). *For any $(n+1)$-type B, the type of functions $H \to B$ is equivalent to*

$$\Sigma\left(f : A \to B\right).\Sigma\left(e : \Pi_{a,b:A}\|a = b\|_{n-1} \to f(a) = f(b)\right).\left(d : \Pi_{a:A}\mathsf{refl}_{f(a)} = e(a, a, |\mathsf{refl}_a|)\right). \tag{41}$$

Proof sketch. The proof of deriving this form of universal property from the induction principle is standard. The map from $H \to B$ into the stated type is more or less composition with the constructors; for any $k : H \to B$, we get

$$(f, e, d) :\equiv \left(k \circ \eta,\ \mathsf{ap}_k \circ \epsilon,\ \lambda a.\mathsf{ap}_{\mathsf{ap}_k}(\delta(a))\right). \tag{42}$$

The map in the other direction is exactly the recursion principle of H. That they are mutually inverse corresponds to the computation (β) rule respectively the uniqueness (η) rule of H.   ◀

Finally, we can complete the second proof of our main result:

"HIT proof" of Lemma 3. We do induction on n. The base case $(n \equiv -1)$ is, as before, just what we have discussed in Section 2. For higher n, we have the following chain of equivalences:

$$\|A\|_n \to B \tag{43}$$

(by Corollary 10)

$$\simeq\ \ H \to B \tag{44}$$

(by Lemma 11)

$$\simeq\ \ \Sigma\left(f : A \to B\right).\Sigma\left(e : \Pi_{a,b:A}\|a = b\|_{n-1} \to f(a) = f(b)\right).$$
$$\left(\Pi_{a:A}\mathsf{refl}_{f(a)} = e(a, a, |\mathsf{refl}_a|)\right) \tag{45}$$

(by "inverse path induction")

$$\simeq \quad \Sigma \left(f : A \to B \right). \Sigma \left(e : \Pi_{a,b:A} \| a = b \|_{n-1} \to f(a) = f(b) \right).$$
$$\left(\Pi_{a,b:A} \Pi_{p:a=b} \mathsf{ap}_f p = e(a, b, |p|) \right) \tag{46}$$

(by the distributivity law)

$$\simeq \quad \Sigma \left(f : A \to B \right). \Pi_{a,b:A} \left(\Sigma \left(e' : \| a = b \|_{n-1} \to f(a) = f(b) \right). \right.$$
$$\left. \Pi_{p:a=b} \mathsf{ap}_f p = e'(|p|) \right) \tag{47}$$

Now we exchange e' by $(e_1, e_2) :\equiv \mathfrak{c}_{n-1}(e')$ using the induction hypothesis, and thus we need to apply $\mathfrak{c}_{n-1}^{-1}$ to that term in the last component. Fortunately, it follows from the definition of $\mathfrak{c}_{n-1}$ that $\_ \circ \mathfrak{c}_{n-1} \equiv \pi_1 \circ |-|$, hence we can replace $e'(|p|)$ with simply $e_1(p)$:

$$\simeq \quad \Sigma \left(f : A \to B \right). \Pi_{a,b:A} \left(\Sigma \left(e_1 : a = b \to f(a) = f(b) \right). \Sigma \left(e_2 : \Pi_{p:a=b} \mathsf{isNull}(\mathsf{ap}_{e_1,p}^n) \right). \right.$$
$$\left. \left(\Pi_{p:a=b} \mathsf{ap}_f p = e_1(p) \right) \right) \tag{48}$$

The term e_1 and the very last (unnamed) component form a singleton and can be removed:

$$\simeq \quad \Sigma \left(f : A \to B \right). \left(\Pi_{a,b:A} \Pi_{p:a=b} \mathsf{isNull}(\mathsf{ap}_{\mathsf{ap}_f,p}^n) \right) \tag{49}$$

(by "path induction")

$$\simeq \quad \Sigma \left(f : A \to B \right). \left(\Pi_{a:A} \mathsf{isNull}(\mathsf{ap}_{\mathsf{ap}_f,\mathsf{refl}_{f(a)}}^n) \right) \tag{50}$$

(as $\mathsf{ap}_{\mathsf{ap}_f,\mathsf{refl}_a}^n$ is the same as $\mathsf{ap}_{f,a}^{n+1}$ – the footnote on page 365 applies)

$$\simeq \quad \Sigma \left(f : A \to B \right). \left(\Pi_{a:A} \mathsf{isNull}(\mathsf{ap}_{f,\mathsf{refl}_a}^{n+1}) \right). \tag{51}$$

Finally, we need to check that the constructed equivalence is indeed the canonical function $\mathfrak{c}_n$. Fortunately, the second (and more involved) part $\Pi_{a:A} \mathsf{isNull}(\mathsf{ap}_{f,\mathsf{refl}_a}^{n+1})$ is propositional. It is therefore enough to check that any map $g : \|A\|_n \to B$ gets, by the constructed equivalence, mapped to a pair in (51) of which *the first component* is $g \circ |-|$. But the first component is constructed in the very first step, where Lemma 11 is applied, and, looking at the proof of Lemma 11, it is indeed simply composition with $|-|$. ◀

5   A Sample Application: Set-Based Groupoids

A set-theoretic ω-groupoid has, in the "globular" formulation, ω-many levels: At level 0, it has a collection of objects (or 0-cells); for any two objects, it has a collection of 1-morphisms (1-cells); for any two 1-morphisms, there is a collection of 2-morphisms (2-cells), and so on. As recalled in the introduction, types indeed are such ω-groupoids meta-theoretically. It is intuitive to ask how much of this can be internalised. Defining a weak ω-groupoid in type theory is already very hard [2, 3]: one would want a 0-type (i.e. a *set*) A_0 of 0-cells, a *set* A_1 of 1-cells which is indexed twice over A_0, and so on. Even if one has such a definition at hand, it is implausible to expect that one can define the "fundamental ω-groupoid" of a type. As Altenkirch, Li and Rypacek [2] mention, they are unable to construct such an ω-groupoid, which in their terminology is called $\mathsf{Id}\omega$. The Ph.D. thesis of the second-named author of the current paper includes a precise negative statement [9, Sec. 9.4.1] which shows that a construction in the sense of [2] is impossible in all non-trivial cases. The argument given

there indicates that a fundamental reason why we cannot even define A_1 is that we want A_1 to be indexed twice over A_0.

However, we know that the whole higher structure of types is in some sense determined by the loop spaces, as opposed to the path spaces. It seems therefore reasonable to consider a more modest variation where we index A_1 only once over A_0, with the intention that $A_1(a_0)$ represents the loop space over a_0. This has the further advantage that we can assume that A_0 is $\|A\|_0$; with double-indexed A_1, it would be possible that elements $a, b : A_0$ are not equal in A_0, but "made equal" by an element of $A_1(a, b)$. As a further simplification, we only consider the question whether a type can be represented in two levels, i.e. with $A_0 \equiv \|A\|_0$ and A_1.

▶ **Definition 12.** We call a type A *set-based representable* if the function

$$\omega_A : A \to \mathcal{U} \tag{52}$$
$$\omega_A(a) :\equiv (a = a) \tag{53}$$

factors through $\|A\|_0$, i.e. if there is a single-indexed family $A_1 : \|A\|_0 \to \mathcal{U}$ of types which, for all $a : A$, satisfies $A_1(|a|) \simeq (a =_A a)$.

We also define the following simple notion:

▶ **Definition 13.** We say that a type A has *loop spaces with braidings* if, for all $a : A$ and $p, q : a = a$, we have $p \cdot q = q \cdot p$.

Examples of types which have loop spaces with braidings are sets (for which the condition is trivial), and, more interestingly, loop spaces themselves.

▶ **Theorem 14.** *Every 1-type whose loop spaces have braidings is set-based representable.*

Proof. As A is a 1-type, the function (52) takes sets as values; that is, in this case, we can assume that ω_A is of type $A \to \mathcal{U}^0$. Using that $\mathcal{U}^0$ is a 1-type [14, Thm. 7.1.11], we may apply Theorem 1 with $n \equiv 0$. We need to show that, for a fixed $a : A$, the function

$$\mathsf{ap}_{\omega_A, a} : \Omega_t(A, a) \to \Omega_t(\mathcal{U}, a = a) \tag{54}$$

is null. But $\mathsf{ap}_{\omega_A}(p)$ induces a *function* of type $(a = a) \to (a = a)$ (via the function that is called $\mathsf{idtoeqv}$ in [14], and projection), and by univalence, it is enough to show that this function does not depend on p. We claim that this function maps $q : a = a$ to $p^{-1} \cdot q \cdot p$. An easy way to prove this claim is considering the more general version of ap_{ω_A} that works on any path spaces (instead of loop spaces), and then doing path induction on p. Clearly, the braiding on $a = a$ is exactly what we need to justify that $p^{-1} \cdot q \cdot p$ does not depend on p. ◀

6 The Big Picture: Solved and Unsolved Cases

The "ordinary" universal property of the n-truncation can be recovered easily from Theorem 1. If, under the conditions of the statement, B is not only $(n+1)$-, but even n-truncated, the type $\Pi_{a:A}\mathsf{isNull}\left(\mathsf{ap}_{f,a}^{n+1}\right)$ becomes contractible, and the theorem says precisely that functions $A \to B$ are the same as functions $\|A\|_n \to B$, via composition with $|-|$. Theorem 1 is thus stronger than the "ordinary" universal property. However, we weaken the condition on B by only one single level, while [8] weakens it by arbitrary many levels, but only for the propositional truncation.

is-?-type(B) / $\|A\|_?$	-1	0	1	2	3	4	$\cdots$	∞
-1		✓ [10]	✓ [8]	✓ [8]	✓ [8]	✓ [8]	$\cdots$	✓ [8]
0			✓ (here)					
1				✓ (here)		unsolved cases		
2		trivial			✓ (here)			
3		– standard universal property applicable				✓ (here)		
$\cdots$								

Figure 2 The universal property of $\|A\|_?$ with respect to ?-types: trivial, solved, and open cases.

Of course, the general question is: What is the universal property of $\|A\|_n$ with respect to m-types, i.e. how can we construct a map $\|A\|_n \to B$ for some m-type B? Put differently, given a function $f : A \to B$, how can we (by only imposing conditions on f, not on A or B) ensure that f factors through $\|A\|_n$? Figure 2 illustrates the current progress on this question. As indicated, the question is trivial if m is not greater than n. Two other families of cases are solved, those with $m \equiv n + 1$ by the current paper, and $n \equiv -1$ by [8]. Note that the latter is not internalised in the way that the result of the current paper is, and it is not to be expected that an internalisation is possible in the considered type theory; and further, the case $n \equiv -1$, $m \equiv \infty$ (meaning that there is no condition at all on B) is solved, but only under the assumption of Reedy ω^{op}-limits.

The (probably) simplest case that is left open is the case $n \equiv 0$, $m \equiv 2$. So, let us consider a function $f : A \to B$, where B is 2-truncated. Which conditions do we have to impose on f to conclude that it factors through $\|A\|_0$? As is easy to show, if f factors through the 0-truncation, then ap_f factors through the (-1)-truncation. The necessary conditions for the latter have been worked out in [8], and we could thus try to impose them on ap_f (at all points). However, this does not work. In one aspect, the propositional truncation is a special case that is actually *harder* than the higher truncations, intuitively because loop spaces are always pointed[10] which we have already made use of in the definition of isNull. It turns out that in this "pointed" case one can get all these coherences (which make the result of [8] hard) for free. Instead, the higher groupoid structure of loop spaces induces a different sort

---

[10] This seems to correspond to the fact that the zeroth homotopy "group" is not a group, and does therefore not have a canonical element, which seems to occasionally make this special case harder in traditional topology as well.

of coherence problem. For example, it certainly *is* necessary that, for any $a : A$ and $p : a = a$, there is a proof $c_{a,p} : \mathsf{ap}_{f,a}(p) = \mathsf{refl}_{f(a)}$. From $c_{a,p}$, we can construct a proof that $\mathsf{ap}_{a,f}(p \cdot p)$ equals $\mathsf{refl}_{f(a)}$, using functoriality of $\mathsf{ap}_{f,a}$. If we want the family c to be "fully coherent", we have to force this proof to be the same as $c_{a,p \cdot p}$. The work [8] concludes with a precise conjecture of how *all* the required coherence conditions can be captured in the general case. At this time, it is unknown whether this can be used to fill in the missing parts of Figure 2.

References

1 Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science (MSCS)*, pages 1–30, Jan 2015.

2 Thorsten Altenkirch, Nuo Li, and Ondrej Rypacek. Some constructions on ω-groupoids. In *Logical Frameworks and Meta-languages: Theory and Practice (LFMTP)*, 2014.

3 Thorsten Altenkirch and Ondrej Rypacek. A syntactical approach to weak ω-groupoids. In *Computer Science Logic (CSL)*, pages 16–30, 2012.

4 Steve Awodey and Andrej Bauer. Propositions as [types]. *Journal of Logic and Computation*, 14(4):447–471, 2004.

5 Paolo Capriotti. Higher lenses. Blog post at `homotopytypetheory.org`, 29 Apr 2014.

6 Paolo Capriotti, Nicolai Kraus, and Andrea Vezzosi. Functions out of higher truncations (Agda formalisation), Apr 2015. Available at `https://github.com/pcapriotti/agda-base/tree/trunc/hott/truncation`.

7 R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the NuPRL Proof Development System.* Prentice-Hall, NJ, 1986.

8 Nicolai Kraus. The general universal property of the propositional truncation. *ArXiv e-prints*, Nov 2014. To appear in the post-proceedings of TYPES'14.

9 Nicolai Kraus. *Truncation Levels in Homotopy Type Theory.* PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2015.

10 Nicolai Kraus, Martín Escardó, Thierry Coquand, and Thorsten Altenkirch. Notions of anonymous existence in Martin-Löf type theory. Submitted, 2014.

11 Nicolai Kraus and Christian Sattler. Higher homotopies in a hierarchy of univalent universes. *ACM Transactions on Computational Logic (TOCL)*, 16(2):18:1–18:12, April 2015.

12 Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. In *Typed Lambda Calculi and Applications (TLCA)*, pages 172–187. Springer-Verlag, 2009.

13 Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, pages 1–75, Jan 2015.

14 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* `homotopytypetheory.org/book`, Institute for Advanced Study, first edition, 2013.

15 Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.