Constructions with Non-Recursive Higher Inductive Types

Nicolai Kraus *
University of Nottingham
nicolai.kraus@nottingham.ac.uk

Abstract

Higher inductive types (HITs) in homotopy type theory are a powerful generalization of inductive types. Not only can they have ordinary constructors to define elements, but also higher constructors to define equalities (paths). We say that a HIT H is *non-recursive* if its constructors do not quantify over elements or paths in H. The advantage of non-recursive HITs is that their elimination principles are easier to apply than those of general HITs.

It is an open question which classes of HITs can be encoded as non-recursive HITs. One result of this paper is the construction of the propositional truncation via a sequence of approximations, yielding a representation as a non-recursive HIT. Compared to a related construction by van Doorn, ours has the advantage that the connectedness level increases in each step, yielding simplified elimination principles into n-types. As the elimination principle of our sequence has strictly lower requirements, we can then prove a similar result for van Doorn's construction. We further derive general elimination principles of higher truncations (say, k-truncations) into n-types, generalizing a previous result by Capriotti et al. which considered the case $n \equiv k + 1$.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic*]: Lambda calculus and related systems

Keywords homotopy type theory, higher inductive types, sequential colimits, truncation elimination, van Doorn construction

1. Introduction

Homotopy type theory, also known as *HoTT*, is a branch of intensional dependent type theory based on the observation that types can be interpreted as (some form of) topological spaces. For a type A with elements $a_1, a_2 : A$, we can view a_1 and a_2 as *points* and the equality type $\mathsf{Id}_A(a_1, a_2)$ (most of the time simply written as $a_1 = a_2$) as the type of *paths* between these points. In the light of HoTT, it is natural to consider a powerful generalization of inductive types, called *higher inductive types (HITs)*, some constructors of which may define elements (*point constructors*) while others may define equalities (*higher constructors*, or *path constructors*). A standard example is the circle $\mathbb{S}^1$, which can be represented as the higher inductive type that is generated by a point constructor $\mathsf{base} : \mathbb{S}^1$ and a path constructor $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$. Another innocent-looking example is the *propositional truncation*: For any type A, the HIT $\|A\|$ is the type generated by a point constructor $|-| : A \to \|A\|$ and a path constructor $\mathsf{t} : \Pi_{u,w:\|A\|}\, u = w$. The propositional truncation is certainly the most prominent concept that can be implemented as a HIT (without being implementable as an ordinary inductive type), as similar operations have been considered long before HoTT was a subject of research. It roughly corresponds to the squash types of NuPRL (Constable et al. 1986) and the concept of bracket types in extensional type theory (Awodey and Bauer 2004). It is interesting because it allows to formulate the *proposition* that a type is inhabited, without the obligation to specify a concrete element of the type. For more detailed examples of HITs, we want to point to the standard reference (Univalent Foundations Program 2013, Chapter 6).

In topology, a particularly nice class of spaces are the *CW complexes*. These can be constructed stepwise. To build a CW complex, one starts with a (discrete) set of points, or 0-cells. For any two points, one may draw a path (or multiple paths) between them; these are the 1-cells. Then, for any configuration of points and lines that forms a cycle, one can attach a 2-cell which has this cycle as its boundary, and so on. A first view on HITs may be that they correspond to CW complexes. This is indeed a good intuition for many basic HITs that are commonly considered: for example, $\mathbb{S}^1$ is really built of one point (the base constructor) and one path (the loop constructor).

However, general HITs are more difficult to understand because they are higher *inductive* types: A constructor of a HIT H may quantify over all elements of H, or even over paths or loops in H, something that one does not allow when constructing CW complexes. Again, the most prominent example is the propositional truncation mentioned above. The first constructor is simple: The point constructor $|-| : A \to \|A\|$ gives us one point for every point in A. However, the path constructor $\mathsf{t} : \Pi_{u,w:\|A\|}\, u = w$ is tricky. It does not correspond to simply adding one path between any two points given by the first constructor. Instead, it means that a path is added between *any* two points of the type that is currently constructed, and not every such point is equal to one that is generated by the first constructor (at least not in a "continuous" way). Intuitively, t also adds paths between points which "lie on paths" that are generated by t itself.

Ordinary inductive types and their expressivity are reasonably well-understood. It is less clear what the status of HITs is. Let us say that a HIT H is *non-recursive* if no constructor quantifies over points or paths in H (a reasonable variant would be to only impose this restrictions on path-constructors, but not on point-constructors, in order to consider an ordinary inductive type to be a non-recursive HIT). At the HoTT workshop in Warsaw (June 29–30, 2015), Altenkirch and the current author have posed the open question whether non-recursive HITs (together with ordinary inductive types) are sufficient to construct all types that can be represented by general HITs. A positive answer would serve as a reduc-

* The author acknowledges support by the Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/M016994/1.

LICS '16, July 05 - 08, 2016, New York, NY, USA

ACM 978-1-4503-4391-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2933575.2933586

tion theorem, similar to the well-known *hubs and spokes* construction (Univalent Foundations Program 2013, Chapter 6.7) which shows that, up to homotopy, all HITs can be constructed using only point- and simple path-constructors, but without constructors for higher paths (i.e. without constructors that define paths between paths). Here, we gloss over the fact that a general satisfactory syntactical scheme for HITs has yet to be established.

If it turned out that any HIT can be constructed as a non-recursive HIT, this would also have practical advantages when using HoTT. The point is that non-recursive HITs have elimination principles that are generally easier to use than those of arbitrary HITs. For example, a map $\mathbb{S}^1 \to X$ for any type X is given by a point $x : X$ and a *loop* (a path whose endpoints coincide) $p : x = x$. In comparison, a map $\|A\| \to X$ is given by a map $A \to X$, but only if X is a propositional type itself (i.e. we need an element of $\Pi_{x,y:X}\, x = y$). This condition on X represents a serious restriction, as it may often happen that one wants a function into some type which is not known to be propositional, in which case a function $f : A \to X$ is not sufficient. Thus, the crucial difference is that general HITs can pose restrictions on the types into which their elimination principles can eliminate, while non-recursive HITs have elimination principles that can be used to eliminate into any type.

The question how to construct functions $\|A\| \to X$ in general has already been examined in previous research, and the goal always is to weaken the requirements on X. The standard reference describes the strategy of finding a propositional type P such that f factors through P (Univalent Foundations Program 2013, Chapter 3.9), while (Kraus et al. 2014) and (Escardó and Xu 2015) describe strategies for several special cases. Previous work by the current author shows that a function $\|A\| \to X$ corresponds to a coherently constant function $A \to X$, which comes with an infinite tower of coherence data, requiring certain Reedy limits. If X is n-truncated for some finite n, then this infinite tower becomes finite, generalizing some of the previously known special cases.

For a given type A, let us consider the HIT $\{A\}$ that is given by a point-constructor $\mathfrak{p} : A \to \{A\}$ and a path-constructor $\mathfrak{e} : \Pi_{a_1,a_2:A}\, \mathfrak{p}(a_1) = \mathfrak{p}(a_2)$. This HIT looks similar to $\|A\|$; however, note that its second constructor does not quantify over elements of $\{A\}$, i.e. it is non-recursive. In fact, it is *very* different from $\|A\|$. While $\|A\|$ is always propositional, $\{A\}$ is never propositional, unless A is empty. For example, in the case that A is the unit type $\mathbf{1}$, the type $\{\mathbf{1}\}$ consists of a point and a loop around that point, which is evidently just $\mathbb{S}^1$. The elimination principle of $\{A\}$ is very simple: a function $\{A\} \to X$ corresponds to a map $f : A \to X$ together with an element of $\Pi_{a_1,a_2} f(a_1) = f(a_2)$; we say that such a function f is *weakly constant*. Note that the terminology *weakly constant* can be somewhat misleading as the weak constancy datum is not well-behaved, and weakly constant functions are not as easy to understand as one could expect (to reformulate the above example: a weakly constant function from $\mathbf{1}$ to X is given by a loop in X). In any case, because of the before-mentioned reason, Altenkirch has called $\{A\}$ the *constant map classifier* in private discussions with the current author. Independently, Coquand and Escardó have called it the *generalized circle* (Coquand and Escardó) because of the connection with $\mathbb{S}^1$ mentioned above.

Moreover, $\{A\}$ plays a central role in a recent construction by van Doorn (van Doorn 2016), who call it the *one-step truncation* as they view $\{A\}$ as a "first step" towards the actual propositional truncation $\|A\|$. More precisely, they consider the sequence

$$A \xrightarrow{\mathfrak{p}} \{A\} \xrightarrow{\mathfrak{p}} \{\{A\}\} \xrightarrow{\mathfrak{p}} \dots \tag{1}$$

and show that the colimit of this sequence, which is a non-recursive HIT, is propositional and has the properties of $\|A\|$. This leads to a new elimination principle for $\|A\|$: a function $\|A\| \to X$ corresponds to a cocone under the sequence (1), that is, a family of functions $\{\dots\{A\}\dots\} \to X$, for any number of applications of $\{-\}$, which are coherent in a certain way. This can be expressed in type theory without additional assumptions.

At first sight, it may be surprising that the colimit of the van Doorn sequence (1) is propositional: Even for simple examples of A (such as the unit type), already $\{\{A\}\}$ and $\{\{\{A\}\}\}$ are very hard to visualize, and there does not seem any aspect in which the sequence becomes simpler. An unpleasant side-effect is that, unlike the elimination principle of (Kraus 2015), van Doorn's elimination principle does not simplify if one wants to eliminate into an n-type; it is still necessary to have an infinite family of functions.

In the current paper, we show that van Doorn's core argument can be generalized: *any* sequence has a propositional colimit, as long as all maps are weakly constant. We sketch a couple of applications of this observation. First of all, the van Doorn construction follows. Further, it allows us to directly see that the ω-sphere, written $\mathbb{S}^\infty$ and constructed as a sequential colimit, is contractible. This fact is well-known, but has so far been proved "manually". We also define a generalized version of the ω-sphere which turns out to be contractible as well.

The main part of the paper examines a generalization of the HIT $\{A\}$, namely the *pseudo-truncation* for any $n \geq -1$, written $(\!|A|\!)_n$. It is derived from the HIT which represents the truncation $\|A\|_n$ by changing the path-constructor so that it only quantifies over elements of A, but not over elements of $(\!|A|\!)_n$. In the same way as $\|A\|$ and $\{A\}$ are different from each other, $\|A\|_n$ and $(\!|A|\!)_n$ are different as well. The connection between them is that we have

$$\|(\!|A|\!)_n\|_{n+1} \;\simeq\; \|A\|_n. \tag{2}$$

This allows us to formulate an elimination principle of the n-truncation into $(k+n)$-types. For $k \equiv 1$, this principle simplifies to (and is proved using) the main result of previous work (Capriotti et al. 2015).

The heart of the paper is our analysis of the sequence

$$A \to (\!|A|\!)_{-1} \to (\!|(\!|A|\!)_{-1}|\!)_0 \to (\!|(\!|(\!|A|\!)_{-1}|\!)_0|\!)_1 \to \dots. \tag{3}$$

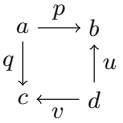
Here, the maps are the canonical maps of the form $\mathfrak{p}_n : A \to (\!|A|\!)_n$. For a type A in general, the only such map which is weakly constant is $\mathfrak{p}_{-1}$; what can be said about all the other maps is much weaker – they are only weakly constant on the $(n+1)$-st higher path spaces. However, assuming $a : A$, we show that in the sequence (1), *each* single map is weakly constant, allowing us to conclude that the sequential colimit is, once more, equivalent to $\|A\|$. Although this construction looks similar to van Doorn's (1), the idea behind the constructions is very different, and so are their consequences. An important feature of (3) is that it is really a sequence of approximations of $\|A\|$ in a suitable sense, and the derived elimination principle becomes finite if one tries to eliminate into an n-type, while (1) seems to be somewhat chaotic. Moreover, we can construct a morphism from our sequence to van Doorn's, implying that *any* cocone of van Doorn's sequence gives a cocone of ours. A particular consequence of this is that the elimination principle from the van Doorn sequence *can* be simplified when one wants to eliminate into an n-type B. For the HIT $\{-\}$, we still do not have the principle (2), and we thus do not have an equivalence between the type $\{\dots\{A\}\dots\} \to B$ with $(n+1)$ iterations of $\{-\}$ and the type $\|A\| \to B$. However, via the morphism of sequences, we can show that these two types are at least logically equivalent (there exist functions in both directions). Thus, a map $\{\dots\{A\}\dots\} \to B$ *does* allow us to construct a map $\|A\| \to B$ if B is n-truncated.

Organization Section 2 very briefly reviews the construction of sequential colimits. In Section 3, we show that the colimit of a sequence of weakly constant maps is propositional, together with a

few applications. What follows in Section 4 is a technical interlude which establishes some connections between loops and maps from spheres mainly using the "adjunction" between the suspension and the loop space operator. In Section 5, we introduce the pseudo-truncation and derive an elimination principle for higher truncations from it. Section 6 contains the proof that every map in (3) is weakly constant, and in Section 7, we derive the mentioned consequence for the van Doorn sequence and compare several elimination principles, and make further concluding remarks.

Setting We work in the type theory that is used in the standard textbook on HoTT (Univalent Foundations Program 2013). We assume familiarity with the constructions and notations that it uses. In particular, we recall that a *pointed type* is a pair of a type A and a point $a : A$, and the type of *pointed maps* $(A, a) \to_\bullet (B, b)$ is defined to be $\Sigma\,(f : A \to B)\,.\,fa = b$. Further, the *suspension* of a type A, written ΣA, is the HIT with two point-constructors north and south, and a family of paths between them which we call the *meridian*, $\mathrm{mrdn} : A \to \mathrm{north} = \mathrm{south}$. To avoid ambiguity when we talk about more than one suspension at the same time, we may write north_A, south_A, and mrdn_A. The sphere $\mathbb{S}^0$ is the two-element type, and $\mathbb{S}^{n+1}$ is by definition $\Sigma\mathbb{S}^n$, meaning that we can view each $\mathbb{S}^n$ as a pointed type $(\mathbb{S}^n, \mathrm{north})$. For any $n \geq -1$ and type A, we have the *n-truncation* $\|A\|_n$, together with a map $|-|_n : A \to \|A\|_n$. In the case $n \equiv -1$, we omit the index and write $|-| : A \to \|A\|$ instead of $|-|_{-1} : A \to \|A\|_{-1}$. If P is a family over a type A, $p : a_1 =_A a_2$ a path in A, and $b_1 : P(a_1)$ and $b_2 : P(a_2)$ points, we use the *path over* notation $b_1 =^P_p b_2$ (Univalent Foundations Program 2013, equation 6.2.2) synonymously to the equality type $p_*(b_1) =_{P(a_2)} b_2$, which in turn reads b_1 *transported along* p *equals* b_2.

Graphical notation We want to emphasize that we use the notations $A \xrightarrow{f} B$ and $a \xrightarrow{p} b$ for both the case of a function $f : A \to B$ and an equality $p : a = b$. As types are notated with capital and elements with lower-case letters, we do not expect a risk of confusion (the notation is not used for type-level equalities). If we say that a diagram commutes, we always mean that it commutes up to homotopy, i.e. that we have an element of an equality type that expresses commutativity. In particular, note that "commuting" is nearly never a propositional property, but an actual datum. As equalities are invertible, we can talk about the composition $a \xrightarrow{p} b \xleftarrow{q} c$. Because of this, we can talk about the commutativity of diagrams such as the one shown on the right. Note that there are many types which express commutativity, e.g. $p \cdot u^{-1} = q \cdot v^{-1}$ or $p \cdot u^{-1} \cdot v \cdot q^{-1} = \mathsf{refl}_a$. As all these types are equivalent, we do not specify which type exactly we mean when we say that the square commutes.



Agda formalization A formalization of the core results is available as an unofficial supplement to the paper (Kraus 2016). This formalization follows the structure of the paper and can be read in parallel, or consulted for specific statements. It builds on the community's HoTT Agda library and benefits greatly from other developers' implementations (see the acknowledgements).

2. Colimits over Graphs

In HoTT, homotopy limits and colimits over graphs are a much simpler concept than homotopy limits and colimits over categories due to the coherence problems involved in the latter. A systematic treatment of homotopy limits over graphs has been presented by (Avigad et al. 2015), which comes with a rigorous formalisation in Coq. Homotopy colimits over graphs have been introduced by (Rijke and Spitters 2014). Here, we only recall the special case that we need, namely colimits over $\mathbb{N}$ (where $\mathbb{N}$ is viewed as a graph with exactly one edge from n to $n+1$ for every n, and no other edges), i.e. colimits of sequences.

We define a *sequence*, more precisely a *sequence of types*, in the obvious way:

Definition 2.1 (Sequence)**.** A *sequence* is a family $A : \mathbb{N} \to \mathcal{U}$ of types, together with a family of functions $f : \Pi_{n:\mathbb{N}}(A_n \to A_{n+1})$. For the sequence given by (A, f), we also write $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots$. Moreover, a *finite sequence* is a pair (A, f) as before, but with $\mathbb{N}$ replaced by a finite set of the form $\{0, 1, \ldots, m-1\}$ (often written as Fin_m). In other words, a finite sequence is simply a finite chain $A_0 \xrightarrow{f_0} \ldots \xrightarrow{f_m} A_m$.

Definition 2.2 (Sequential colimit)**.** We define the *sequential colimit* of a sequence $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots$ to be the higher inductive type A_ω with the two constructors $\mathfrak{i}$ (*"insert"*) and $\mathfrak{g}$ (*"glue"*) as follows:

- $\mathfrak{i} : \Pi_{n:\mathbb{N}}\,(A_n \to A_\omega)$
- $\mathfrak{g} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} \mathfrak{i}_n a =_{A_\omega} \mathfrak{i}_{n+1}(f_n a)$.

The induction principle of the sequential colimit is straightforward to write down:

Principle 2.3 (Sequential colimit - induction)**.** For a given sequence $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots$, the induction principle of A_ω is given as follows. Assume $P : A_\omega \to \mathcal{U}$ is a type family. Assume further that we have a pair $(\bar{\mathfrak{i}}, \bar{\mathfrak{g}})$ of terms of the following types:

- $\bar{\mathfrak{i}} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} P(\mathfrak{i}_n a)$
- $\bar{\mathfrak{g}} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} \bar{\mathfrak{i}}_n a =^P_{\mathfrak{g}_n a} \bar{\mathfrak{i}}_{n+1}(f_n a)$.

Then, there is a term

$$\mathsf{ind}^{A_\omega}_{\bar{\mathfrak{i}},\bar{\mathfrak{g}}} : \Pi_{x:A_\omega} P(x) \tag{4}$$

with the judgmental computation rule $\mathsf{ind}^{A_\omega}_{\bar{\mathfrak{i}},\bar{\mathfrak{g}}}(\mathfrak{i}_n a) \equiv \bar{\mathfrak{i}}_n a$ as well as the "homotopy" computation rule $\mathsf{apd}_{\mathsf{ind}^{A_\omega}_{\bar{\mathfrak{i}},\bar{\mathfrak{g}}}}(\mathfrak{g}_n a) = \bar{\mathfrak{g}}_n a$.

The following is straightforward, and we record it for future use. We refer to our formalization for a rigorous proof.

Lemma 2.4. *The colimit of a sequence is equivalent to the colimit of the sequence with a finite initial segment removed. That is, for a sequence $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots$, the colimit A_ω is equivalent to the colimit of the sequence $A_n \xrightarrow{f_n} A_{n+1} \xrightarrow{f_{n+1}} \ldots$.* □

The following notation will be useful at several points in the paper:

Notation 2.5. Let k and m be natural numbers with $k < m$. Given a sequence $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots$, we write $f^m_k : A_i \to A_{m+1}$ for the composition $f_m \circ f_{m-1} \circ \ldots \circ f_k$. If we are given a point $a : A_k$, we can consider a sequence of equalities in A_ω, namely

$$\mathfrak{i}_k a \xrightarrow{\mathfrak{g}_k a} \mathfrak{i}_{k+1}(f_k a) \xrightarrow{\mathfrak{g}_{k+1}(f_k a)} \ldots \xrightarrow{\mathfrak{g}_m(f^{m-1}_k a)} \mathfrak{i}_{m+1}(f^m_k a). \tag{5}$$

We write $\mathfrak{g}^m_k a : \mathfrak{i}_k a = \mathfrak{i}_{m+1}(f^m_k a)$ for this composition.

3. Weakly Constant Sequences

We say that a function $f : A \to B$ is *weakly constant* if it maps any two elements to equal values:

$$\mathsf{wconst}(f) :\equiv \Pi_{a_1,a_2:A} f(a_1) = f(a_2). \tag{6}$$

By a theorem of (van Doorn 2016), the colimit of the sequence $A \to \{A\} \to \{\{A\}\} \to \ldots$ is propositional, where $\{A\}$ is van

Doorn's *one-step truncation* (for this result, see Example 3.2 below, and for a generalization of the type operator $\{-\}$, see Section 5). We show that this result holds for *any* sequence, as long as all the maps are weakly constant. The main steps of the proofs are the same as in van Doorn's proof. However, for this generalization, we crucially rely on a strategy that simplified Doorn's original proof, suggested by the current author (van Doorn, blog post comment section).

Lemma 3.1. *Assume we are given a sequence $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \dots$ and every f_i is weakly constant. Then, A_ω is propositional.*

Proof. By assumption, we have $c_k : \mathsf{wconst}(f_k)$ for every k. Note that this means that $\mathsf{i}_k : A_k \to A_\omega$ is weakly constant as well, as for any $a, a' : A_k$ we have

$$\mathsf{i}_k a \xrightarrow{\mathfrak{g}_k a} \mathsf{i}_{k+1}(f_k a) \xrightarrow{\mathsf{ap}_{\mathsf{i}_{k+1}}(c_k(a,a'))} \mathsf{i}_{k+1}(f_k a') \xleftarrow{\mathfrak{g}_k a'} \mathsf{i}_k a'. \quad (7)$$

To prove the lemma, it is sufficient to show $A_\omega \to \mathsf{isContr}(A_\omega)$. By the recursion principle of the sequential colimit (the non-dependent version of the induction principle), this means that we need to construct

$$\tilde{\mathsf{i}} : \Pi_{n:\mathbb{N}}(A_n \to \mathsf{isContr}(A_\omega)) \quad (8)$$

$$\tilde{\mathsf{g}} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} \tilde{\mathsf{i}}_n a = \tilde{\mathsf{i}}_{n+1}(f_n a). \quad (9)$$

Since the type of $\tilde{\mathsf{g}}$ is contractible, we only need $\tilde{\mathsf{i}}$. Let us fix $n : \mathbb{N}$; we need to show $A_n \to \mathsf{isContr}(A_\omega)$. By Lemma 2.4, A_ω is contractible if and only if the colimit of $A_n \xrightarrow{f_n} A_{n+1} \xrightarrow{f_{n+1}} \dots$ is, so we may prove this instead. By re-indexing, we can ensure $n \equiv 0$, and we may thus assume $a_0 : A_0$.

We choose $\mathsf{i}_0 a_0 : A_\omega$ as the center of contraction, and therefore, we need to construct an element of $\Pi_{w:A_\omega} P(w)$ with

$$P : A_\omega \to \mathcal{U} \quad (10)$$

$$P(w) :\equiv w = \mathsf{i}_0(a_0). \quad (11)$$

We do induction on w, i.e. we apply Principle 2.3 a second time. Thus, we need $\bar{\mathsf{i}}$ and $\overline{\mathsf{g}}$, the types of which are

$$\bar{\mathsf{i}} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} \mathsf{i}_n a = \mathsf{i}_0(a_0) \quad (12)$$

$$\overline{\mathsf{g}} : \Pi_{n:\mathbb{N}} \Pi_{a:A_n} \bar{\mathsf{i}}_n a =^P_{\mathfrak{g}_n a} \bar{\mathsf{i}}_{n+1}(f_n a). \quad (13)$$

The type of $\overline{\mathsf{g}}_n a$ is, by an application of a standard lemma (Univalent Foundations Program 2013, Theorem 2.11.3), equivalent to

$$\bar{\mathsf{i}}_n a \;=\; \mathfrak{g}_n a \cdot \bar{\mathsf{i}}_{n+1}(f_n a). \quad (14)$$

We construct $\bar{\mathsf{i}}_n a$ as the composition:

$$\bar{\mathsf{i}}_n a :\equiv \mathfrak{g}_n a \cdot \mathsf{ap}_{\mathsf{i}_{n+1}}(c_n(a, f_0^{n-1} a_0)) \cdot \mathfrak{g}_0^n(a_0)^{-1}. \quad (15)$$

We want to remind the reader of Notation 2.5: f_0^n is a composition of functions, while $\mathfrak{g}_0^n$ is a concatenation of equalities. Note that the proof constructed in (15) is the concatenation of the above proof that i_n is weakly constant with $\mathfrak{g}_0^{n-1}(a_0)$.

The construction of (13) requires more work. Let us consider Figure 1. By (14), what we need to show is the commutativity of the triangle built of the dashed arrows and the arrow labelled $\mathfrak{g}_n a$. The two quadrangles labelled ① and ② commute by the construction of $\bar{\mathsf{i}}_n a$. Hence, we need to show that the heptagon of solid arrows commutes.

Our strategy is to simplify the solid heptagon until we see that it commutes. For our next step, let us look at Figure 2 which shows the heptagon again. Some of the heptagon's faces are dashed, and some additional arrows are added. The triangles ③ and ④ commute trivially. The two parallel arrows ⑤ are equal because i_{n+2} is weakly constant, implying that $\mathsf{ap}_{\mathsf{i}_{n+2}}$ is constant as well (Kraus et al. 2013, 2014).

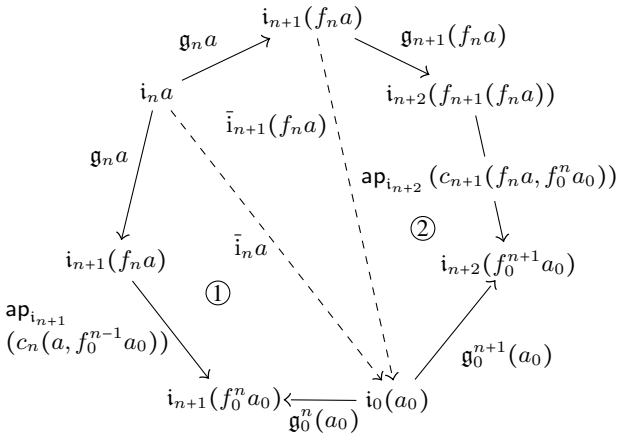


Figure 1: Construction of $\overline{\mathsf{g}}$ (13), first step

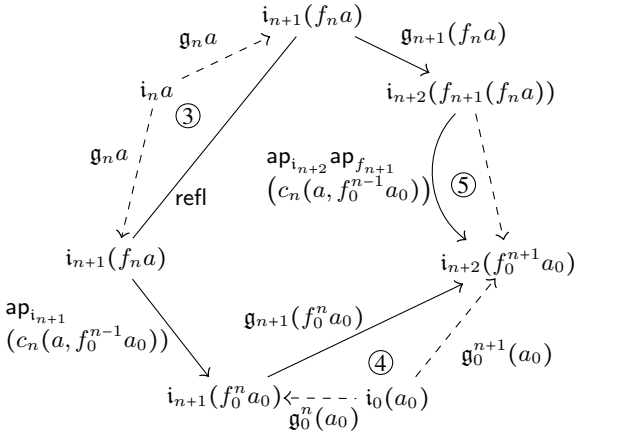


Figure 2: Construction of $\overline{\mathsf{g}}$ (13), second step

Therefore, we are left with proving that the solid pentagon in Figure 2 commutes. This becomes easy when we generalize the situation: Let us replace $f_n a$ by some $x : A_{n+1}$, and $f_0^n a_0$ by some $y : A_{n+1}$, and $c_n(a, f_0^{n-1} a_0)$ by some proof $q : x = y$. What we need to prove becomes

$$\mathfrak{g}_{n+1}(x) \cdot \left(\mathsf{ap}_{\mathsf{i}_{n+2}} \mathsf{ap}_{f_{n+1}}(q)\right) \;=\; \mathsf{ap}_{\mathsf{i}_{n+1}}(q) \cdot \mathfrak{g}_{n+1}(y), \quad (16)$$

and this is obvious by induction on q. □

Sample Applications In the remainder of this section, we demonstrate a few applications of Lemma 3.1. These will not be required for our main results.

Example 3.2 (The construction of $\|-\|$ by (van Doorn 2016))**.** As discussed in the introduction, van Doorn defines the *one-step truncation* of a type A, written $\{A\}$, as the HIT with the constructors

- $\mathfrak{p} : A \to \{A\}$
- $\mathfrak{e} : \Pi_{a_1,a_2:A} f(a_1) = f(a_2)$.

They then consider the sequence

$$A \xrightarrow{\mathfrak{p}} \{A\} \xrightarrow{\mathfrak{p}} \{\{A\}\} \xrightarrow{\mathfrak{p}} \dots \quad (17)$$

and show that the colimit $\{A\}^\omega$ has all the properties of $\|A\|$, which means that $\|-\|$ can be constructed using only non-recursive HITs. Let us reconstruct this result.

We write $\{A\}^n$ for $\{\dots\{A\}\dots\}$ with n applications of $\{-\}$. The counterpart of $|-| : A \to \|A\|$ is the map $\mathrm{i}_0 : A \to \{A\}^\omega$. Next, we need to show that for any propositional type P, the map $(\{A\}^\omega \to P) \to (A \to P)$ given by composition with i_0 is an equivalence. As both function types are propositional, it is sufficient to construct *any* function in the other direction. Assume we are given $f : A \to P$. We need to construct a map $\{A\}^\omega \to P$. By recursion on the sequential colimit, we need a family of functions $\bar{\mathrm{i}}_n : \{A\}^n \to P$; the coherence cells $\bar{\mathrm{g}}$ are automatic as equalities in propositional types. We do induction on n. The map $\bar{\mathrm{i}}_0$ is given by f. For a function $\bar{\mathrm{i}}_{n+1} : \{\{A\}^n\} \to P$, we apply recursion on the one-step truncation. We need to provide a map $\{A\}^n \to P$, which is given by $\bar{\mathrm{i}}_n$, and we need to show $\Pi_{a_1,a_2:\{A\}^n} \bar{\mathrm{i}}_n a_1 = \bar{\mathrm{i}}_n a_2$, which is again automatic. The judgmental computation rule is inherited from the one of the colimit.

The hard part of van Doorn's construction is to show that $\{A\}^\omega$ is propositional. This is a direct consequence of Lemma 3.1, as the constructor $\mathfrak{e}$ ensures that each map $\mathfrak{p}$ is weakly constant.

Example 3.3 ($\mathbb{S}^\infty$ is contractible)**.** Let us quickly recall that the suspension operator Σ is functorial: given a function $f : A \to B$, we get a function $\Sigma f : \Sigma A \to \Sigma B$. Concretely, Σf is defined by Σ-recursion: we map north_A to north_B and south_A to south_B, and the required function $A \to \mathrm{north}_B = \mathrm{south}_B$ is given my $\mathrm{mrdn}_B \circ f$.

We further note that, assuming $a_0 : A$ and weak constancy of f, the map Σf is constantly north_B and thus weakly constant as well. Let us show $Q :\equiv \Pi_{x:\Sigma A} \Sigma f(x) = \mathrm{north}_B$ by Σ-induction. To do this, we need elements $\overline{n} : Q(\mathrm{north}_A)$, and $\overline{s} : Q(\mathrm{south}_A)$, and finally $\overline{m} : \Pi_{a:A} \overline{n} =^Q_{\mathrm{mrdn}_A(a)} \overline{s}$. We define $\overline{n}$ to be reflexivity and $\overline{s}$ to be $\mathrm{mrdn}_B(a_0)$. By standard calculations (Univalent Foundations Program 2013, Theorem 2.11.3), the type of $\overline{m}(a)$ becomes $\mathsf{ap}_{\Sigma f}(\mathrm{mrdn}_A a) = \mathrm{mrdn}_B(f a_0)$. We have the "homotopy computation rule" $\mathsf{ap}_{\Sigma f}(\mathrm{mrdn}_A a) = \mathrm{mrdn}_B(f a)$, and the required equation follows from the fact that f is weakly constant.

The ∞-sphere can be constructed in at least two reasonable ways, as indicated in the standard reference (Univalent Foundations Program 2013, Exercise 8.3 and 8.4). One possibility is to construct it as the sequential colimit of the sequence

$$\mathbb{S}^0 \xrightarrow{f_0} \mathbb{S}^1 \xrightarrow{f_1} \mathbb{S}^2 \xrightarrow{f_2} \dots, \tag{18}$$

where the maps f_n are defined by induction on n. The function f_0 simply maps the two points of $\mathbb{S}^0$ to north and south, respectively. Further, we define $f_{n+1} :\equiv \Sigma(f_n)$.

The function f_0 is clearly weakly constant, and thus every map f_n by what we have established above. Thus, $\mathbb{S}^\infty$ is propositional by Lemma 3.1 and, as it is inhabited by $\mathrm{i}_1(\mathrm{north})$, it is contractible.

There is an alternative way of defining the suspension which we call the *equatorial suspension*, written Σ^e. The feature of this version is that there is an "equator" constructor which directly gives a map $A \to \Sigma^e A$. The situation is illustrated in Figure 3.

Definition 3.4 (equatorial suspension)**.** For a type A we define the *equatorial suspension* $\Sigma^e A$ as the higher inductive type with the constructors

- $\mathrm{north} : \Sigma^e A$
- $\mathrm{south} : \Sigma^e A$
- $\mathrm{eqtr} : A \to \Sigma^e A$
- $\mathrm{n} : \Pi_{a:A} \mathrm{north} = \mathrm{eqtr}(a)$
- $\mathrm{s} : \Pi_{a:A} \mathrm{eqtr}(a) = \mathrm{south}$.

Lemma 3.5. *The equatorial suspension is equivalent to the ordinary suspension. That is, for a type A, we have $\Sigma^e A \simeq \Sigma A$.*

Proof. This equivalence is straightforward, although the precise formalization via the induction principles is tedious. The key observation is that the pair $(\mathrm{eqtr}, \mathrm{n})$ forms a singleton type, implying that the type $\Sigma^e A$ has the same universal property as ΣA. □

Remark 3.6. Of course, under the equivalence sketched in the proof of Lemma 3.5 above, the map $\mathrm{eqtr} : A \to \Sigma^e A$ simply becomes the map $A \to \Sigma A$ which is constantly north.

Example 3.7 (The generalized ∞-sphere is contractible)**.** Let A be a type. We can consider the sequence

$$A \xrightarrow{\mathrm{eqtr}} \Sigma^e A \xrightarrow{\mathrm{eqtr}} \Sigma^e (\Sigma^e A) \xrightarrow{\mathrm{eqtr}} \dots. \tag{19}$$

We may call the colimit of this sequence the *generalized ∞-sphere* because, for $A \equiv \mathbb{S}^0$, the sequence (19) becomes equal to the sequence (18). To see this, we simply need to use that by what we said in Example 3.3, the maps in (18) are all constantly north, and compare this to Remark 3.6.

This generalized ∞-sphere is contractible, not matter what A is, by Lemma 3.1.

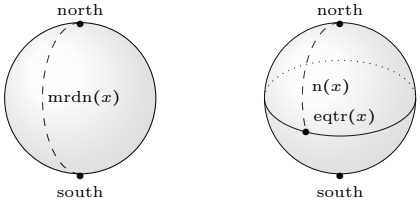


Figure 3: The 2-sphere as suspension (left) and as equatorial suspension (right)

4. A Technical Interlude: The Correspondance between Loops and Maps from Spheres

For the further development, we need to work out several technical statements. The core ingredient of these observations is the well-known fact that there morally is an adjunction $\Sigma \dashv \Omega$ between the suspension and the loop space "functor". If one settles for an appropriate notion of $(\infty, 1)$-*category*, the author expects that this can be turned into a precise statement. Here, we choose to work on a lower level and manually prove some consequences of the conjectured adjunction.

To begin with, recall the following result from (Univalent Foundations Program 2013, Lemma 6.5.4):

Lemma 4.1. *For pointed types A and B, there is a map*

$$\Phi_{A,B} : (\Sigma A \to_\bullet B) \to (A \to_\bullet \Omega B) \tag{20}$$

which is an equivalence. □

We will usually omit the type indices of the function (20) and simply write Φ instead of $\Phi_{A,B}$. Note that the type of pointed maps $(X, x_0) \to_\bullet (Y, y_0)$ has always a canonical element, namely $\mathbf{c}_{y_0} :\equiv (\lambda x.y_0, \mathsf{refl}_{y_0})$. The following to lemmata are easy to verify by analyzing how Φ is constructed; we refer to the Agda formalization for the proofs. The first lemma states that Φ is a pointed map itself:

Lemma 4.2. *For any (A, a_0) and (B, b_0), the map* (20) *preserves the canonical element in the sense that $\Phi(\mathbf{c}_{b_0}) = \mathbf{c}_{\mathsf{refl}_{b_0}}$.* □

The second lemma expresses naturality of the "hom-set isomorphism" Φ in the second argument (naturality in the first argument

is analogous, but we will not need it). Note that it is standard in homotopy type theory to write ap_g instead of Ωg for a map g between pointed types. We further write $\mathsf{ap}_g^\bullet$ for the pointed version of ap_g (carrying the obvious proof that refl is preserved).

Lemma 4.3 (Naturality of Φ in second argument)**.** *For pointed types and maps as in $\Sigma A \xrightarrow{f}_\bullet B \xrightarrow{g}_\bullet C$, the equation*

$$\Phi(g \circ f) =_{A \to_\bullet \Omega C} \mathsf{ap}_g^\bullet \circ \Phi(f). \tag{21}$$

holds. □

Given a function $f : X \to Y$ and a point $y_0 : Y$, we can say that f *is null* (Capriotti et al. 2015) if

$$\mathsf{isNull}(f) :\equiv \Pi_{x:X} f(x) = y_0. \tag{22}$$

We can also talk about $\mathsf{isNull}(f)$ if f is a pointed map, in which case we simply mean that the underlying map is null with respect to the point of the codomain. Alternatively, we can extend the notion and define, for a pointed map $g : (A, a_0) \to_\bullet (B, b_0)$,

$$\mathsf{isNull}^\bullet(g) :\equiv g = \mathbf{c}_{b_0}. \tag{23}$$

As a small caveat we want to remark that "being null" is not in general a propositional property in either case.

The connection between (22) and (23) is the following:

Lemma 4.4. *If $f : A \to_\bullet B$, is a pointed map, then we have the logical equivalence*

$$\mathsf{isNull}(f) \longleftrightarrow \mathsf{isNull}^\bullet(f). \tag{24}$$

Proof. The direction "←" is obvious. For the other direction, assume we are given f, p, and a proof of $\mathsf{isNull}(f)$ in the form of an element $q : \Pi_{a:A} f(a) = b_0$. The term $q' :\equiv \lambda a.q(a) \cdot q(a_0)^{-1} \cdot p$ (which is of the same type as q) satisfies $q'(a_0) = p$, allowing us to construct an element of $\mathsf{isNull}^\bullet(f)$. □

Lemma 4.5. *For a pointed map $g : \Sigma(A, a_0) \to_\bullet (B, b_0)$, we have*

$$\mathsf{isNull}^\bullet(g) \simeq \mathsf{isNull}^\bullet(\Phi(g)). \tag{25}$$

Proof. This follows directly from the definition (23), Lemma 4.2, and the fact that equivalences preserve path spaces. □

Remark 4.6. For a pointed map $(f, p) : \Sigma(A, a_0) \to_\bullet (B, b_0)$, we have

$$\mathsf{isNull}(f) \longleftrightarrow \mathsf{isNull}(\mathsf{fst}(\Phi(f, p))) \tag{26}$$

by combining Lemma 4.4 and Lemma 4.5. Note that it cannot be strengthened to an actual (homotopy) equivalence. If B is an $(n+2)$-type, then $\mathsf{isNull}(f)$ is an $(n+1)$-type (and not always an n-type), while $\mathsf{isNull}(\mathsf{fst}(\Phi(f, p)))$ is always n-truncated.

For a pointed map $g : B \to_\bullet C$, we can iterate $\mathsf{ap}^\bullet$ to construct a function

$$\mathsf{ap}_g^{\bullet\, m} : \Omega^m(B) \to_\bullet \Omega^m(C). \tag{27}$$

Lemma 4.7. *Let k and m be natural numbers, and assume that $\mathbb{S}^{k+m} \xrightarrow{f}_\bullet B \xrightarrow{g}_\bullet C$ are two pointed functions. We may then also consider the composition $\mathbb{S}^k \xrightarrow{\Phi^m(f)}_\bullet \Omega^m(B) \xrightarrow{\mathsf{ap}_g^{\bullet\, m}}_\bullet \Omega^m(C)$. We have an equivalence*

$$\mathsf{isNull}^\bullet(g \circ f) \simeq \mathsf{isNull}(\mathsf{ap}_g^{\bullet\, m} \circ \Phi^m(f)). \tag{28}$$

Proof. Induction on m using the Lemmata 4.3 and 4.5. □

We have two ways of expressing that a function is null on path-level m. The following lemma, which arises as a special case of the previous statement, connects these.

Lemma 4.8. *Let $m > 0$ be a number and $g : (B, b_0) \to_\bullet (C, c_0)$ be a pointed function. Then, we have*

$$\left(\Pi_{f:\mathbb{S}^m \to_\bullet (B,b_0)} \mathsf{isNull}^\bullet(g \circ f)\right) \simeq \mathsf{isNull}(\mathsf{ap}_g^{\bullet\, m}). \tag{29}$$

Proof. This is Lemma 4.7 with $k \equiv 0$ and the observation that $\mathbb{S}^0 \to_\bullet X$ is equivalent to X for any pointed type X. □

A further useful consequence is the following:

Lemma 4.9. *Let n be a natural number and $P : \mathbb{S}^{n+1} \to \mathcal{U}$ be a family of types such that $P(\mathrm{north})$ is $(n-1)$-truncated. Then, we can construct a function*

$$P(\mathrm{north}) \to \Pi_{y:\mathbb{S}^{n+1}} P(y). \tag{30}$$

Proof. We regard P as a pointed map $\mathbb{S}^{n+1} \to_\bullet (\mathcal{U}, P(\mathrm{north}))$. Consider the type $\Omega^{n+1}(\mathcal{U}, P(\mathrm{north}))$. By (Kraus and Sattler 2015, Lemma 5.2), it equals $\Pi_{z:P(\mathrm{north})} \Omega^n(P(\mathrm{north}), z)$, which in turn is contractible as $P(\mathrm{north})$ is $(n-1)$-truncated (Univalent Foundations Program 2013, Theorem 7.2.9). This shows that $\Phi^{n+1}(P, \mathsf{refl})$ is null, and so is P by Lemma 4.7 (where the second map is simply the identity). Hence, $P = \lambda y.P(\mathrm{north})$, and the claimed map (30) is trivial to construct. □

5. General Pseudo-Truncations

In this section, we will generalize the HIT $\{-\}$ and prove some basic results about this generalization. Let us start with the definition of the n-truncation $\|A\|_n$ as it is given in (Univalent Foundations Program 2013, Chapter 7.3). It is a HIT with a constructor $|-|_n : A \to \|A\|_n$; for every function $r : \mathbb{S}^{n+1} \to \|A\|_n$, a *hub* $\mathsf{h}(r) : \|A\|_n$; and, for every r as before and every $x : \mathbb{S}^{n+1}$, a *spoke* path $\mathsf{s}_r(x) : r(x) = \mathsf{h}(r)$. We change the constructors so that they only quantify over maps $\mathbb{S}^{n+1} \to A$ instead of maps $\mathbb{S}^{n+1} \to \|A\|_n$, making sure that the resulting HIT is presented non-recursively, and we call this HIT the *pseudo-n-truncation*. Of course, this HIT will usually not be equivalent to the actual n-truncation, but there are some connections which we will examine later.

Definition 5.1 (Pseudo-truncation)**.** For a number $n \geq -1$ and a type A, the *n-th pseudo-truncation* of A is a higher inductive type $(\!|A|\!)_n$ with the three constructors $\mathfrak{p}_n$ (*"points"*), $\mathfrak{h}_n$ (*"hubs"*), and $\mathfrak{s}_n$ (*"spokes"*), as follows:

- $\mathfrak{p}_n : A \to (\!|A|\!)_n$
- $\mathfrak{h}_n : \left(\mathbb{S}^{n+1} \to A\right) \to (\!|A|\!)_n$
- $\mathfrak{s}_n : \Pi_{r:\mathbb{S}^{n+1} \to A} \Pi_{x:\mathbb{S}^{n+1}} \mathfrak{p}_n(rx) = \mathfrak{h}_n(r)$.

The pseudo-truncation $(\!|-|\!)_-$ is to be understood as a family that is parametrized over both a suitably defined type of numbers and the relevant type universe, in the same way as the truncation $\|-\|_-$ is. Let us give its induction principle:

Principle 5.2 (Pseudo-truncation – induction)**.** Given n and A, the induction principle of $(\!|A|\!)_n$ is the following. Assume we have a family $P : (\!|A|\!)_n \to \mathcal{U}$, and terms $(\overline{\mathfrak{p}}_n, \overline{\mathfrak{h}}_n, \overline{\mathfrak{s}}_n)$ of the following types:

- $\overline{\mathfrak{p}}_n : (a : A) \to P(\mathfrak{p}_n(a))$
- $\overline{\mathfrak{h}}_n : (r : \mathbb{S}^{n+1} \to A) \to P(\mathfrak{h}_n(r))$
- $\overline{\mathfrak{s}}_n : (r : \mathbb{S}^{n+1} \to A) \to (x : \mathbb{S}^{n+1}) \to \overline{\mathfrak{p}}_n(rx) =^P_{\mathfrak{s}_n(r)} \overline{\mathfrak{h}}_n(r)$,

Then, there is a term

$$\mathsf{ind}^{(\!|-|\!)_n}_{\overline{\mathfrak{p}}_n, \overline{\mathfrak{h}}_n, \overline{\mathfrak{s}}_n} : \Pi_{x:(\!|A|\!)_n} P(x). \tag{31}$$

Moreover, this term satisfies the judgmental computation rules

$$\mathsf{ind}^{(\!|-|\!)_n}_{\overline{\mathfrak{p}}_n, \overline{\mathfrak{h}}_n, \overline{\mathfrak{s}}_n}(\mathfrak{p}_n(a)) \equiv \overline{\mathfrak{p}}_n(a) \tag{32}$$

and

$$\mathsf{ind}^{(\!|-|\!)_n}_{\overline{\mathrm{p}}_n,\overline{\mathrm{h}}_n,\overline{\mathrm{s}}_n}(\mathfrak{h}_n(r)) \;\equiv\; \overline{\mathrm{h}}_n(r) \tag{33}$$

as well as the "homotopy computation rule"

$$\mathsf{ap}_{\mathsf{ind}^{(\!|-|\!)_n}_{\overline{\mathrm{p}}_n,\overline{\mathrm{h}}_n,\overline{\mathrm{s}}_n}}(\mathfrak{s}_n(r,x)) \;=\; \overline{\mathrm{s}}_n(r,x). \tag{34}$$

Remark 5.3. Regarding the above definition, we want to note two things.

1. It is easy to check that $(\!|A|\!)_{-1}$ is equivalent to van Doorn's one-step truncation $\{A\}$ that has been discussion in the introduction and in Example 3.2.
2. In the definition of the pseudo-n-truncation, the hub constructor $\mathfrak{h}_n$ could be removed if we let $\mathfrak{s}_n$ construct a path $\mathfrak{p}_n(rx) = \mathfrak{p}_n(r(\mathrm{north}))$ instead. For $n > -1$, the resulting HIT would in general not be equivalent to the HIT that we have defined, for the reason explained by (Univalent Foundations Program 2013, Remark 6.7.1). However, the resulting HIT would behave very similar, and we expect that all the further results of this paper would hold for this modification of $(\!|-|\!)_n$ as well.

The induction principle of the pseudo-truncation is powerful enough to emulate the induction principle of the "real" truncation:

Lemma 5.4. *Let A be a type and n be a number, as well as $P : (\!|A|\!)_n \to \mathcal{U}$ a family of n-types. Then, we have the "weak induction" principle*

$$\mathsf{ind}^{(\!|-|\!)_n}_{\mathsf{w}} : (\Pi_{a:A} P(\mathfrak{p}_n a)) \to \Pi_{x:(\!|A|\!)_n} P(x) \tag{35}$$

such that $\mathsf{ind}^{(\!|-|\!)_n}_{\mathsf{w}}(f) \circ \mathfrak{p}_n \equiv f$.

Proof. We assume that we are given $f : \Pi_{a:A} P(\mathfrak{p}_n a)$, and we do induction on $x : (\!|A|\!)_n$. We choose

$$\overline{\mathrm{p}}_n :\equiv f \tag{36}$$

(which already ensures the claimed judgmental equality) and

$$\overline{\mathrm{h}}_n(r) :\equiv \mathsf{transport}^P\left((\mathfrak{s}_n(r,\mathrm{north}))\,,\, f(r(\mathrm{north}))\right). \tag{37}$$

Finally, we need to define $\overline{\mathrm{s}}_n(r)$, the type of which has to be

$$\Pi_{y:\mathbb{S}^{n+1}} \overline{\mathrm{p}}_n(ry) =^P_{\mathfrak{s}_n(r)} \overline{\mathrm{h}}_n(r). \tag{38}$$

By the assumption that P is a family of n-types, the type family here is $(n-1)$-truncated, and by Lemma 4.9, it is enough to construct an inhabitant for $y \equiv \mathrm{north}$. Simplifying the $\mathsf{transport}$-terms, we see that this equality type is canonically inhabited. □

Recall Notation 2.5: given a sequence, we write f^m_k for the concatenation $f_m \circ \ldots f_k$. For any n, we consider the n-th pseudo-truncation to be an endofunction on the universe (which is how HITs are commonly understood); that is, we have $(\!|-|\!)_n : \mathcal{U} \to \mathcal{U}$. Therefore, for given numbers $k < m$, we write $(\!|-|\!)^m_k$ for the composition $(\!|\ldots(\!|-|\!)_k\ldots|\!)_m$.

Further, if A is a given type and $k \geq -1$ a number, we can consider the sequence

$$A \xrightarrow{\mathfrak{p}_k} (\!|A|\!)_k \xrightarrow{\mathfrak{p}_{k+1}} (\!|A|\!)^{k+1}_k \xrightarrow{\mathfrak{p}_{k+2}} (\!|A|\!)^{k+2}_k \xrightarrow{\mathfrak{p}_{k+3}} \ldots, \tag{39}$$

and we write $(\!|A|\!)^\omega_k$ for its sequential colimit. Our interest will lie on $(\!|A|\!)^\omega_{-1}$. The following is a derived induction principle of $(\!|A|\!)^\omega_k$ which shows that this colimit satisfies the elimination rule of $\|A\|_k$.

Lemma 5.5 (derived induction principle for $(\!|-|\!)^\omega_k$)**.** *Let A be a type and $P : (\!|A|\!)^\omega_k \to \mathcal{U}$ a family of k-types, for some $k \geq -1$. Then, we can derive*

$$\mathsf{ind}^{(\!|-|\!)^\omega_n}_{\mathsf{d}} : (\Pi_{a:A} P(\mathfrak{i}_0 a)) \to \Pi_{x:(\!|A|\!)^\omega_k} P(x). \tag{40}$$

such that $\mathsf{ind}^{(\!|-|\!)^\omega_n}_{\mathsf{d}}(f) \circ \mathfrak{i}_0 \equiv f$.

The special case where P is a constant family (i.e. P is simply $\lambda a.Q$ for some type Q gives a recursion principle: for a k-type Q such that $A \to Q$, we get $(\!|A|\!)^\omega_k \to Q$.

Proof of Lemma 5.5. This is an advanced version of the argument given in Example 3.2. Assume we are given $f : \Pi_{a:A} P(\mathfrak{i}_0 a)$. We first do induction on the sequential colimit. This means we need to construct a family $f_m : \Pi_{x:(\!|A|\!)^{k+m-1}_k} P(\mathfrak{i}_m x)$ (note that we use the notation $(\!|A|\!)^{k-1}_k \equiv A$). We choose $f_0 :\equiv f$, ensuring that the claimed judgmental equality is satisfied. Next, we can construct f_{n+1} from f_n as follows. We consider $Q_{n+1} :\equiv P \circ \mathfrak{i}_{n+1}$. In this formulation, what we need is $f_{n+1} : \Pi_{x:(\!|A|\!)^{k+n}_k} Q(x)$. We see that f_n is exactly what we need to use Lemma 5.4, and the equality stated in that lemma gives us the coherences that we need between f_n and f_{n+1}. □

To conclude the section, let us establish a few direct connections between the pseudo-truncation and the truncation.

Lemma 5.6. *For a type A and $n \geq -1$, the map $|-|_n : A \to \|A\|_n$ factors through $(\!|A|\!)_n$. That is, there is a map $u : (\!|A|\!)_n \to \|A\|_n$ such that $|-|_n = u \circ \mathfrak{p}_n$.*

Proof. This is given by the special case $P :\equiv \lambda x.\|A\|_n$ of the "weak induction principle" in Lemma 5.4. □

A central result about pseudo-truncations is the following:

Theorem 5.7. *For a type A and a number $n \geq -1$, we have*

$$\|(\!|A|\!)_n\|_{n+1} \;\simeq\; \|A\|_n. \tag{41}$$

Before giving the proof, we state an immediate consequence:

Corollary 5.8. *Let A be a type and k, m be numbers, $-1 \leq k \leq m$. Then, we have the equivalence*

$$\left\|(\!|A|\!)^{m-1}_k\right\|_m \;\simeq\; \|A\|_k. \tag{42}$$

Consequently, if B is an m-type, then we have

$$\left((\!|A|\!)^{m-1}_k \to B\right) \;\simeq\; \left(\|A\|_k \to B\right). \tag{43}$$

This gives an elimination principle for k-truncations into m-types for all finite numbers k and m (note that the case $k \geq m$ is trivial).

Proof of Theorem 5.7. Firstly, the identity on $\|(\!|X|\!)_n\|_{n+1}$ factors through $\|X\|_n$. To show this, let us write f for the composition

$$A \xrightarrow{\mathfrak{p}_n} (\!|A|\!)_n \xrightarrow{|-|_{n+1}} \|(\!|X|\!)_n\|_{n+1} \xrightarrow{\mathsf{id}} \|(\!|X|\!)_n\|_{n+1}. \tag{44}$$

Clearly, $\|(\!|X|\!)_n\|_{n+1}$ is an $(n+1)$-type. By (Capriotti et al. 2015, Theorem 1), f factors through $\|A\|_n$ if (and only if), for every $a : A$, the map $\mathsf{ap}^{n+1}_{f,a} : \Omega^{n+1}(A,a) \to \Omega^{n+1}\left(\|(\!|A|\!)_n\|_{n+1}, |\mathfrak{p}_n a|_{n+1}\right)$ is null. This is guaranteed by the constructors $\mathfrak{h}_n$ and $\mathfrak{s}_n$ together with Lemma 4.8. Therefore, we get the map $t : \|A\|_n \to \|(\!|A|\!)_n\|_{n+1}$ as shown in Figure 4, and the big shape commutes. The rest is a "diagram chase" in Figure 4. We get the map u and commutativity of ① by Lemma 5.6. As any map into an n-type factors through the $(n+1)$-truncation, we get s and that ② commutes. Applying the induction principle of the truncation, we know that ③ commutes if (and only if) $|-|_{n+1} = t \circ s \circ |-|_{n+1}$. By Lemma 5.4, we can compose each side with $\mathfrak{p}_n$ and we get the equation $|-|_{n+1} \circ \mathfrak{p}_n = t \circ s \circ |-|_{n+1} \circ \mathfrak{p}_n$ which holds as ②, ① and the big shape commute. This shows $t \circ s = \mathsf{id}$.

To prove $s \circ t = \mathsf{id}$, we can compose both sides with $|-|_n$ by the induction principle of $\|-\|_n$. But $s \circ t \circ |-|_n = |-|_n$ using commutativity of the big shape, ②, and ① in this order. □

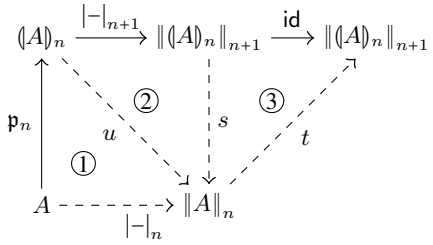


Figure 4: Factoring the identity through $\|A\|_n$

6. The Propositional Truncation as Non-Recursive HIT

Let A be a type, and let us consider the sequence

$$A \xrightarrow{\mathfrak{p}_{-1}} (\!|A|\!)_{-1} \xrightarrow{\mathfrak{p}_0} (\!|A|\!)^0_{-1} \xrightarrow{\mathfrak{p}_1} (\!|A|\!)^1_{-1} \xrightarrow{\mathfrak{p}_2} \dots. \tag{45}$$

The goal of the current is to show that the colimit $(\!|A|\!)^\infty_{-1}$ has all the properties of the propositional truncation $\|A\|$, and thus represents a non-recursive construction of the propositional truncation.

From Corollary 5.8, we have the equivalence

$$\left\|(\!|A|\!)^{m-1}_{-1}\right\|_m \;\simeq\; \|A\| \tag{46}$$

for any m. Because of this, we say that $(\!|A|\!)^{m-1}_{-1}$ is *conditionally m-connected* (it is m-connected if it is inhabited). From this, it is not hard to see that the the sequential colimit $(\!|A|\!)^\infty_{-1}$ is also conditionally m-connected, for any number m with $-1 \leq m < \infty$. Unfortunately, this does not entail that $(\!|A|\!)^\infty_{-1}$ is propositional itself. The problem is that Whitehead's principle is not provable in HoTT (Univalent Foundations Program 2013, Chapter 8.8). If we have a type X and we know that X is m-connected for any finite number m, we cannot conclude that X is contractible. Therefore, the result of Section 5 is not sufficient to conclude that $(\!|A|\!)^\infty_{-1}$ is propositional.

In Section 3, we have established the result that the colimit of a sequence is propositional if all the maps are weakly constant. Of course, the map $\mathfrak{p}_{-1} : A \to (\!|A|\!)_{-1}$ is weakly constant. However, for $n \geq 0$, the map $\mathfrak{p}_n : A \to (\!|A|\!)_n$ satisfies only a much weaker condition that can be phrased as "constancy on $(n+1)$-st path spaces" (see Lemma 4.8). In particular, $\mathfrak{p}_n$ is usually not weakly constant, as the following two observations show:

1. If a_1 and a_2 are not equal in A, then $\mathfrak{p}_n a_1$ and $\mathfrak{p}_n a_2$ are not equal in $(\!|A|\!)_n$.
2. If $\mathfrak{p}_n$ is weakly constant, then $|-|_n$ is also weakly constant (which happens exactly if A is conditionally n-connected) using Lemma 5.6.

We do not know whether the second point can be reversed. We conjecture that this is *not* the case; it looks as if one would need to make some non-trivial choice to do this (note that, if A is conditionally n-connected, then the statement "the map $|-|_n : A \to \|A\|_n$ is weakly constant" is propositional, but the statement "the map $\mathfrak{p}_n$ is weakly constant" is not necessarily propositional).

If the second point could actually be reversed, it could indeed be used to show that each map in the sequence (45) is weakly constant as we have already established in the previous paragraph that $(\!|A|\!)^{m-1}_{-1}$ is conditionally m-connected. However, this would still not be satisfactory: we want to show that $(\!|A|\!)^\infty_{-1}$ represents a construction of the propositional truncation using only non-recursive HITs, and the higher truncation operators $\|-\|_n$ that we are using are implemented as recursive HITs.

Our proof for the fact that $(\!|A|\!)^\infty_{-1}$ is propositional does go via Lemma 3.1, i.e. the result that the colimit of a chain of weakly constant functions is propositional. However, we do not show this for any type A, but only for pointed A; and we do think that this assumption is necessary. Fortunately, it will afterwards turn out that this assumption is unproblematic.

Lemma 6.1. *Let A be a type with a point $a : A$. Then, every map in the sequence*

$$A \xrightarrow{\mathfrak{p}_{-1}} (\!|A|\!)_{-1} \xrightarrow{\mathfrak{p}_0} (\!|A|\!)^0_{-1} \xrightarrow{\mathfrak{p}_1} (\!|A|\!)^1_{-1} \xrightarrow{\mathfrak{p}_2} \dots \tag{47}$$

is weakly constant.

Proof of Lemma 6.1. For every $j \geq -2$, and every $y : (\!|A|\!)^j_{-1}$, we will construct a term

$$c_{j,y} : \mathfrak{p}_{j+1}(y) =_{(\!|A|\!)^{j+1}_{-1}} \mathfrak{p}^{j+1}_{-1}(a). \tag{48}$$

Of course, the concatenation of $c_{j,y}$ and ${c_{j,y'}}^{-1}$ will then be of the required type $\mathfrak{p}_{j+1}(y) = \mathfrak{p}_{j+1}(y')$. To construct $c_{j,y}$, we do induction on j.

Case $\mathbf{j \equiv -2}$ For $y : A$ we need to show $\mathfrak{p}_{-1}(y) =_{(\!|A|\!)_{-1}} \mathfrak{p}_{-1}(a)$. Recall from Remark 5.3 that the constructors $\mathfrak{h}_{-1}$ and $\mathfrak{s}_{-1}$ essentially say that $\mathfrak{p}_{-1}$ is weakly constant. This makes this case very easy. In detail, recall that $\mathbb{S}^0$ is a type with two elements, say north and south. Consider the function $r : \mathbb{S}^0 \to A$ mapping north to y and south to a. We define $c_{-2,y}$ to be the composition $\mathfrak{s}_{-1}(r, \mathrm{north}) \cdot (\mathfrak{s}_{-1}(r, \mathrm{south}))^{-1}$.

Case $j \equiv i+1$ We want to do induction on $y : (\!|A|\!)^j_{-1}$. As the considered type family is given by

$$P(y) :\equiv \mathfrak{p}_{j+1}(y) =_{(\!|A|\!)^{j+1}_{-1}} \mathfrak{p}^{j+1}_{-1}(a), \tag{49}$$

the data that we need has the types

$$\overline{\mathrm{p}}_j : \Pi_{w:(\!|A|\!)^{j-1}_{-1}} \mathfrak{p}_{j+1}(\mathfrak{p}_j(w)) =_{(\!|A|\!)^{j+1}_{-1}} \mathfrak{p}^{j+1}_{-1}(a) \tag{50}$$

$$\overline{\mathrm{h}}_j : \Pi_{r:\mathbb{S}^{j+1}\to(\!|A|\!)^{j-1}_{-1}} \mathfrak{p}_{j+1}(\mathfrak{h}_j(r)) =_{(\!|A|\!)^{j+1}_{-1}} \mathfrak{p}^{j+1}_{-1}(a) \tag{51}$$

$$\overline{\mathrm{s}}_j : \Pi_{r:\mathbb{S}^{j+1}\to(\!|A|\!)^{j-1}_{-1}} \Pi_{x:\mathbb{S}^{j+1}} \overline{\mathrm{p}}_j(rx) =^P_{\mathfrak{s}_j(r)} \overline{\mathrm{h}}_j(r). \tag{52}$$

For $\overline{\mathrm{p}}_j$, we choose $\mathsf{ap}_{\mathfrak{p}_{j+1}}$ applied on the induction hypothesis,

$$\overline{\mathrm{p}}_j(w) :\equiv \mathsf{ap}_{\mathfrak{p}_{j+1}}(c_{j-1,w}). \tag{53}$$

Next, we choose

$$\overline{\mathrm{h}}_j(r) :\equiv \mathsf{ap}_{\mathfrak{p}_{j+1}}\left((\mathfrak{s}_j(r, \mathrm{north}))^{-1} \cdot c_{j-1,r(\mathrm{north})}\right). \tag{54}$$

To construct $\overline{\mathrm{s}}_j$, let us start by fixing a function $r : \mathbb{S}^{j+1} \to (\!|A|\!)^{j-1}_{-1}$. For every $x : \mathbb{S}^{j+1}$ we need to show

$$\overline{\mathrm{p}}_j(rx) =^P_{\mathfrak{s}_j(r)} \overline{\mathrm{h}}_j(r). \tag{55}$$

By a standard lemma (Univalent Foundations Program 2013, Theorem 2.11.3), the type (55) expresses commutativity of the following triangle:

$$\begin{array}{ccc} \mathfrak{p}_{j+1}(\mathfrak{p}_j(rx)) & \xrightarrow{\mathsf{ap}_{\mathfrak{p}_{j+1}}(\mathfrak{s}_j(r,x))} & \mathfrak{p}_{j+1}(\mathfrak{h}_j r) \\ & {\scriptstyle \overline{\mathrm{p}}_j(rx)} \searrow \quad \swarrow {\scriptstyle \overline{\mathrm{h}}_j(r)} & \\ & \mathfrak{p}^{j+1}_{-1}(a) & \end{array} \tag{56}$$

For every $x : \mathbb{S}^{j+1}$, this triangle can be viewed as a loop k_x based at $\mathfrak{p}^{j+1}_{-1}(a)$, and we need to show that each k_x is equal to refl.

The core observation of this proof is that k_x can be written as $\mathsf{ap}_{\mathfrak{p}_{j+1}}(h_x)$, where h is given by

$$h : \mathbb{S}^{j+1} \to \mathfrak{p}^j_{-1}(a) = \mathfrak{p}^j_{-1}(a) \tag{57}$$

$$h_x :\equiv (c_{j-1,rx})^{-1} \cdot \mathfrak{s}_j(r,x) \cdot (\mathfrak{s}_j(r,\mathrm{north}))^{-1} \cdot c_{j-1,r(\mathrm{north})} \tag{58}$$

Is is easy to see that h_{north} equals refl. We can thus view h as a *pointed* map

$$h : \mathbb{S}^{j+1} \to_\bullet \Omega\left(|A|^j_{-1}, \mathfrak{p}^j_{-1}(a)\right). \tag{59}$$

We need to show that $\mathsf{ap}^\bullet_{\mathfrak{p}_{j+1}} \circ h$ is null, and by Lemma 4.7, this is the case if the composition

$$\mathbb{S}^{j+2} \xrightarrow{\Phi^{-1}(h)}_\bullet \left(|A|^j_{-1}, \mathfrak{p}^j_{-1}(a)\right) \xrightarrow{\mathfrak{p}_{j+1}}_\bullet \left(|A|^{j+1}_{-1}, \mathfrak{p}^{j+1}_{-1}(a)\right) \tag{60}$$

is null ($\mathfrak{p}_{j+1}$ is viewed as a pointed map in the obvious way). By Lemma 4.4, it is enough to show that the underlying composition $\mathbb{S}^{j+2} \to |A|^j_{-1} \xrightarrow{\mathfrak{p}_{j+1}} |A|^{j+1}_{-1}$ maps every $x : \mathbb{S}^{j+2}$ to the same point as it maps north to, which is guaranteed by the constructors $\mathfrak{h}_{j+1}$ and $\mathfrak{s}_{j+1}$. This completes the construction of $\overline{\mathfrak{s}}_j$. □

The just proved lemma is the main ingredient for the proof of the result of the current section:

Theorem 6.2. *In HoTT, the type $|A|^\infty_{-1}$ is equivalent to the propositional truncation $\|A\|_{-1}$. More precisely, in HoTT without general recursive HITs, but only non-recursive HITs, the type $|A|^\infty_{-1}$ has all the properties that one expects of the propositional truncation of A, i.e. the propositional truncation can be constructed.*

Proof. The map $\mathfrak{i}_0 : A \to |A|^\infty_{-1}$ plays the role of $|-| : A \to \|A\|_{-1}$. From Lemma 5.5, we have the correct induction and recursion principles, including the judgmental computation rules. We need to show that $|A|^\infty_{-1}$ is propositional. To do this, it is enough to show that any $z : |A|^\infty_{-1}$ implies that $|A|^\infty_{-1}$ is contractible. Using Lemma 5.5, we can assume that z is $\mathfrak{i}_0(a)$ with some $a : A$, and we may simply concatenate the Lemmata 6.1 and 3.1. □

The above theorem implies immediately that functions out of $\|-\|$ are equivalent to cocones under the sequence that we consider:

Corollary 6.3. *For types A and B, a function $g : \|A\| \to B$ corresponds to a family of functions $f_n : |A|^{n+1}_{-1} \to B$ which is coherent in the sense that $f_n = f_{n+1} \circ \mathfrak{p}_{n+1}$.* □

7. Conclusions: On Elimination Principles of Truncations

Apart from a construction of $\|-\|$ via non-recursive HITs, the main presented results are characterizations of function spaces $\|A\|_k \to B$, where B is either assumed to be m-truncated as in Corollary 5.8 or an arbitrary type as in Corollary 6.3. We use this section to compare these results with those of previous articles.

Consequences for the van Doorn sequence With the results by van Doorn and the current paper, we have two sequences which have the propositional truncation as their colimit. Van Doorn's sequence can, with the notation as in Example 3.2, be written as

$$A \xrightarrow{\mathfrak{p}} \{A\} \xrightarrow{\mathfrak{p}} \{A\}^2 \xrightarrow{\mathfrak{p}} \dots. \tag{61}$$

Note that $\{-\}$ could equivalently be replaced by $(\!|-|\!)_{-1}$ by Remark 5.3. The sequence discussed in the current paper is

$$A \xrightarrow{\mathfrak{p}_{-1}} |A|_{-1} \xrightarrow{\mathfrak{p}_0} |A|^0_{-1} \xrightarrow{\mathfrak{p}_1} \dots. \tag{62}$$

In each case, it follows that we can construct functions $\|A\| \to B$ by giving a cocone under the sequence, and we can ask how these cocones compare to each other.

Of course, the type of cocones under (61) is equivalent to the type of cocones under (62), as both correspond to $\|A\| \to B$. At the same time, our sequence has the advantage that the finite initial segments are better behaved: If B is an m-truncated type, then the "full" type of cocones under our sequence is equivalent to a finite initial segment, namely the cocones under $A \to \dots \to |A|^{m-1}_{-1}$; and such a cocone is of course determined by a single map $|A|^{m-1}_{-1} \to B$. This is captured as the special case $k \equiv -1$ by Corollary 5.8. Nothing similar is true for the van Doorn sequence: even if B is m-truncated (with $m \geq 0$), maps $\|A\| \to B$ can only be described as cocones under the full sequence, but not under a finite initial segment.

Moreover, our sequence is less "demanding" than van Doorn's, in the sense that it is generally easier to construct a cocone under our sequence than under van Doorn's; this can be summarized by proving that there is a "natural transformation" from our sequence to van Doorn's:

Theorem 7.1. *For types A and B, there is an $\mathbb{N}$-indexed family of functions $g_n : |A|^{n-2}_{-1} \to \{A\}^n$ such that each square of the form*

$$\begin{array}{ccc} |A|^{n-2}_{-1} & \xrightarrow{\mathfrak{p}_{n-1}} & |A|^{n-1}_{-1} \\ {\scriptstyle g_n}\downarrow & & \downarrow{\scriptstyle g_{n+1}} \\ \{A\}^n & \xrightarrow[\mathfrak{p}]{} & \{A\}^{n+1} \end{array}$$

commutes.

Proof. The core idea behind this statement is that, if a map is weakly constant (such as the constructor $\mathfrak{p}$), then it is weakly constant on higher loop spaces (such as the constructor $\mathfrak{p}_n$). A precise proof proceeds as follows. For any type C, we can construct a map $k_{n+1} : (\!|C|\!)_n \to \{C\}$ via the recursion principle of $(\!|-|\!)_n$ (see Principle 5.2). As always, we need to construct three components: First, a map $C \to \{C\}$; we simply use $\mathfrak{p}$. Second, for any $r : \mathbb{S}^{n+1} \to C$ a *hub* point; we take $\mathfrak{p}(r(\mathrm{north})) : \{C\}$. Third, for r as before and $x : \mathbb{S}^{n+1}$, we need to construct $\mathfrak{p}(r(x)) = \mathfrak{p}(r(\mathrm{north}))$; but this is immediately given by the second constructor of $\{C\}$.

The map g_0 is trivial. Given g_n, we get g_{n+1} as the composition

$$|A|^{n-1}_{-1} \xrightarrow{(\!|g_n|\!)_{n-1}} (\!|\{A\}^n|\!)_{n-1} \xrightarrow{k_n} \{A\}^{n+1}, \tag{63}$$

where we use that $(\!|-|\!)_{-1}$ is ("homotopy") functorial. □

A nice consequence for the van Doorn sequence is that we *can* use a finite initial segment of the infinite cocone to eliminate into an m-type (again, such a finite initial segment is determined by a single map):

Corollary 7.2. *Given an m-type B and a type A. Then, we have a logical equivalence*

$$\left(\{A\}^{n+1} \to B\right) \longleftrightarrow \left(\|A\| \to B\right). \tag{64}$$

Proof. The direction "←" is trivial, and "→" is done as follows. Given a function $f : \{A\}^{n+1} \to B$, we compose it with $g_{n+1} : |A|^{n-1}_{-1} \to \{A\}^{n+1}$ from Theorem 7.1. Then, we apply the second function from Corollary 5.8 (with $k :\equiv -1$ and $m :\equiv n$). □

Functions out of Higher Truncations Corollary 5.8 enables us to construct functions $\|A\|_k \to B$, if B is an m-type. This generalizes (Capriotti et al. 2015) which considers the case $m \equiv k+1$.

Universal Properties of the Propositional Truncation It is worth comparing the characterizations of functions $\|A\| \to B$ via cocones over van Doorn's and our sequence, as analyzed above, to the "general universal property of the propositional truncation" given

in (Kraus 2015). In that work, functions $\|A\| \to B$ are shown to be equivalent to *coherently constant* functions. These consists of an infinite tower of coherence data, similar as the discussed cocones have an $\mathbb{N}$-indexed family of components. On the first levels, a coherently constant function consists of:

1. a function $f : A \to B$. This is the same as the first component of both van Doorn's and our cocones.
2. a proof c that f is weakly constant. This is still the same as for the cocones.
3. a proof that c is coherent in the sense of $c_{x,y} \cdot c_{y,z} = c_{x,z}$. This is much more minimalistic than the condition that the cocones encode. It says that any triangle generated by c can be filled. In the case of our sequence, we require that *any* triangle (the type of triangles is equivalent to the type of loops) can be filled, not matter whether it was generated by previous constructors or whether it already existed anyway, while van Doorn's sequence demands that *any* two points are equal, no matter where they come from, an even stronger requirement.

From here, the data of coherently constant functions diverges considerately from (and is much easier to satisfy than) what the cocones encode. In total, it seems that the coherently constant functions of (Kraus 2015) give much more minimalistic characterizations of maps out of the propositional truncation. This is easy to see if one tries to unfold what the requirements for the cocones are without the nice syntax that the HITs offer. On the other hand, one requires the type theory to have certain Reedy limits in order to even state the type of coherently constant functions, while the characterization via cocones is completely internal.

Final Conclusions Although the construction of the propositional truncation presented in the current paper looks similar to the one presented by van Doorn, the two ideas behind the sequences are rather different. Van Doorn's idea is to use the *one-step truncation* in order to make any two existing points equal. This creates a chaos of new paths which do not behave well in any sense. This chaos is cleaned up in the next step, which in turn creates even more chaos (as $\{A\}$ is usually more complicated than A), and so on. In the colimit, there is no "last step", and hence no remaining chaos. In comparison, our sequence (62) tries to approximate $\|A\|$ stepwise. In the first step, we create paths between any two points in order to ensure that the result will be conditionally 0-truncated. In the second step, we fill all "open 1-loops" in order to get a conditionally 1-truncated type, and in the n-th step, we fill all "open loops" on level $(n-1)$. This is much more similar to the idea of *coherently constant* functions in (Kraus 2015), and Altenkirch and the current author have considered the sequence (62) before learning about van Doorn's result; however, the crucial fact that the proof of the truncation level of the colimit factors through Lemma 3.1 is inspired by van Doorn's construction.

What is left open is whether the type $(\!|A|\!)^\omega_k$ is k-truncated and represents the k-truncation of A in general. This seems likely but would require a generalization of the developed techniques. For this reason, we have only derived a characterization of maps from $\|A\|_k$ into m-types, but not into arbitrary types. It also seems likely that $\|A\|_k$ can be constructed via the sequence consisting *only* of iterations of $(\!|-|\!)_k$ along the lines of van Doorn's construction, which however would again give a weaker elimination principle.

Finally, the main question motivating further research in the direction pursued in the current paper is of course the following question, originally posed as an open problem by Altenkirch and the current author at the HoTT workshop in Warsaw (June 29–30, 2015): Can all HITs be represented as non-recursive HITs, or for which classes of HITs is this the case?

Acknowledgments

I am grateful to Thorsten Altenkirch (from whom I initially learnt the idea of iterating non-recursive truncations) for several fruitful discussions on the topic, and to Floris van Doorn for explanations on the one-step truncation. I would like to thank all contributors of the HoTT Agda library, in particular Guillaume Brunerie, Evan Cavallo, and Kuen-Bang Hou (Favonia). Especially the implementation of the Σ-Ω adjunction by Evan Cavallo has been extremely useful when I mechanized the core results of this paper in Agda.

References

J. Avigad, K. Kapulkin, and P. L. Lumsdaine. Homotopy limits in type theory. *Mathematical Structures in Computer Science*, 25(05):1040–1070, 2015. doi: 10.1017/S0960129514000498.

S. Awodey and A. Bauer. Propositions as [types]. *Journal of Logic and Computation*, 14(4):447–471, 2004. doi: 10.1093/logcom/14.4.447.

P. Capriotti, N. Kraus, and A. Vezzosi. Functions out of higher truncations. *24th EACSL Annual Conference on Computer Science Logic (CSL'15)*, volume 41 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 359–373, 2015. ISBN 978-3-939897-90-3. doi: 10.4230/LIPIcs.CSL.2015.359.

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. ISBN 0-13-451832-2.

T. Coquand and M. H. Escardó. The geometry of constancy. Unpublished note, presented at the Warsaw workshop on HoTT/UF, 30 June 2015.

M. H. Escardó and C. Xu. The Inconsistency of a Brouwerian Continuity Principle with the CurryâĂŞHoward Interpretation. *13th International Conference on Typed Lambda Calculi and Applications (TLCA'15)*, volume 38 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 153–164, 2015. ISBN 978-3-939897-87-3. doi: 10.4230/LIPIcs.TLCA.2015.153.

N. Kraus. The general universal property of the propositional truncation. *20th International Conference on Types for Proofs and Programs (TYPES'14)*, volume 39 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111–145, 2015. ISBN 978-3-939897-88-0. doi: 10.4230/LIPIcs.TYPES.2014.111.

N. Kraus. Agda formalization: Constructions with non-recursive higher inductive types, 2016. Unofficial supplement of the paper, available on GitHub (`github.com/nicolaikraus/HoTT-Agda`) and the author's institutional webpage.

N. Kraus and C. Sattler. Higher homotopies in a hierarchy of univalent universes. *ACM Transactions on Computational Logic (TOCL)*, 16(2): 18:1–18:12, April 2015. doi: 10.1145/2729979.

N. Kraus, M. Escardó, T. Coquand, and T. Altenkirch. Generalizations of Hedberg's theorem. *Typed Lambda Calculus and Applications (TLCA'13)*, volume 7941 of *Lecture Notes in Computer Science*, pages 173–188. Springer-Verlag, 2013. doi: 10.1007/978-3-642-38946-7_14.

N. Kraus, M. Escardó, T. Coquand, and T. Altenkirch. Notions of anonymous existence in Martin-Löf type theory. Submitted to the special issue of TLCA'13, 2014.

E. Rijke and B. Spitters. Sets in homotopy type theory. *MSCS, special issue: From type theory and homotopy theory to univalent foundations*, 2014. doi: 10.1017/S0960129514000553.

T. Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. homotopytypetheory.org/book, Institute for Advanced Study, 2013.

F. van Doorn. Constructing the propositional truncation using nonrecursive hits. Blog post at `homotopytypetheory.org/2015/07/28`.

F. van Doorn. Constructing the propositional truncation using nonrecursive hits. In *5th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP'16)*, pages 122–129, 2016. doi: 10.1145/2854065.2854076.