NOTIONS OF ANONYMOUS EXISTENCE IN MARTIN-LÖF TYPE THEORY

NICOLAI KRAUS, MARTÍN ESCARDÓ, THIERRY COQUAND, AND THORSTEN ALTENKIRCH

University of Nottingham, School of Computer Science, Wollaton Road, Nottingham NG8 1BB, UK
e-mail address: psxngk@nottingham.ac.uk

University of Birmingham, School of Computer Science, Birmingham B15 2TT, UK
e-mail address: m.escardo@cs.bham.ac.uk

Chalmers University, Department of Computer Science and Engineering, SE-412 96 Göteborg, Sweden
e-mail address: Thierry.Coquand@cse.gu.se

University of Nottingham, School of Computer Science, Wollaton Road, Nottingham NG8 1BB, UK
e-mail address: thorsten.altenkirch@nottingham.ac.uk

Abstract. As the groupoid model of Hofmann and Streicher proves, identity proofs in intensional Martin-Löf Type Theory can not generally shown to be unique. Inspired by a theorem by Hedberg, we give some simple characterizations of types that do have unique identity proofs. A key ingredient in these constructions are weakly constant endofunctions on identity types. We study such endofunctions on arbitrary types and show that they always factor through a propositional type, the *truncated* or *squashed* domain. Followed by a discussion on why this is not possible for weakly constant functions in general, we present certain non-trivial cases in which it can be done. Our results also enable us to define a new notion of anonymous existence in type theory, and different forms of existence are carefully compared. In addition, we show possibly surprising consequences of the judgmental computation rule of the truncation, in particular in the context of homotopy type theory.

All the results have been formalized and verified in the dependently typed programming language Agda.

1998 ACM Subject Classification: F.4.1 [Mathematical Logic and Formal Languages] - Lambda calculus and related systems.

Key words and phrases: homotopy type theory, Hedberg's theorem, propositional equality, anonymous existence, constant functions, factorization, truncation, squash types, bracket types, coherence conditions.

Part of this article appeared as "Generalizations of Hedberg's Theorem" in the proceedings of Typed Lambda Calculus and Applications 2013.

Supported by the ERC project 247219, and grants of The Ellentuck and The Simonyi Fund.

Supported by the EPSRC grant EP/G03298X/1 and by a grant of the Institute for Advanced Study.

1. Introduction

Although the identity type is defined as an inductive type with only one single constructor refl, it is arguable the hardest concept in Martin-Löf Type Theory [19] [16] [17] [18] to get intuition for. The reason is that it is, as a type family, parametrized twice over the same type, while the constructor only expects one argument: $\mathsf{refl}_a : a = a$, where the latter is an alternative notation for $\mathsf{Id}(a, a)$. In fact, it is the simplest and most natural occurrence of this phenomenon.

A surprising result by Hofmann and Streicher [10] is that we can not prove refl_a to be the only inhabitant of the type $a = a$, that is, the principle of *unique identity proofs* (UIP) is not derivable. Some time later, Hedberg [9] formulated a sufficient condition on a type to satisfy UIP, namely that its equality is decidable.

It took more than ten years before a more semantic explanation of the Hofmann-Streicher discovery was found. Awodey and Warren [3] as well as, independently, Voevodsky [29] explained that types can be regarded as, roughly speaking, topological spaces. Consequently, an exciting new direction of constructive formal mathematics attracted researchers from originally very separated areas of mathematics, and *Homotopy Type Theory* [27] was born.

The current article is not only on Homotopy Type Theory, but on Martin-Löf Type Theory in general, even though we expect that the results are most interesting in the context of Homotopy Type Theory. We start with Hedberg's Theorem [9] and describe multiple simple ways of strengthening it, one of them involving *propositional truncation* [27], also known as *bracket types* [4] or *squash types* [6].

Propositional truncation is a concept that provides a sequel to the well-known *Propositions as Types* paradigm [11] [18]. If we regard a type as the correspondent of a mathematical statement, a *proposition*, and its inhabitants of proofs thereof, we have to notice that there is a slightly unsatisfactory aspect. A proof of a proposition in Mathematics is usually not thought to contain any information apart from the fact that the proposition is true; however, a type can have any number of inhabitants, and therefore any number of witnesses of its truth. Hence it seems natural to regard only *some* types as propositions, namely those which have at most one inhabitant. The notion of propositional truncation assigns to any type the proposition that this type is inhabited. To make the connection clearer, these types are even called *propositions*, or *h-propositions*, in Homotopy Type Theory. With this in mind, we want to be able to say that a type is inhabited without having to reveal an inhabitant explicitly. This is exactly what propositional truncation $\|-\| : \mathcal{U} \to \mathcal{U}$ (where we write $\mathcal{U}$ for the universe of types) makes possible. On the other hand, should A have only one inhabitant up to the internal equality, this inhabitant can be constructed from an inhabitant of $\|A\|$. This is a crucial difference between propositional truncation and double negation. We consider a weak version of $\|-\|$ which does not have judgmental computation properties.

After discussing direct generalizations of Hedberg's Theorem, we attempt to transfer the results from the original setting, where they talk about equality types (of *path spaces*), to arbitrary types. This leads to a broad discussion of *weakly constant functions*: we say that $f : A \to B$ is *weakly constant* if it maps any two elements of A to equal elements of B. The attribute *weakly* comes from the fact that we do not require these actual equality proofs to fulfill further conditions, and a weakly constant function does not necessarily appear to be constant in the topological models. For exactly this reason, it is in general not possible

to factorize the function f through $\|A\|$; however, we can do it in certain special cases, and we analyze why. This has, for example, the consequence that the truncated sum of two proposition already has the universal property of their *join*, which is defined as a *higher inductive type* in Homotopy Type Theory.

Particularly interesting are weakly constant *endofunctions*. We show that these can always be factorized through the propositional truncation, based on the observation that the type of fixed points of such a function is a proposition. This allows us to define a new notion of existence which we call *populatedness*. We say that A is populated if any weakly constant endofunction on A has a fixed point. This property is propositional and behaves very similar to $\|A\|$, but we show that it is strictly weaker. On the other hand, it is strictly stronger than the double negation $\neg\neg A$, another notion of existence which, however, is often not useful as it generally only allows to prove negative statements. It is worth emphasizing that our populatedness is not a component that has to be *added* to type theory, but a notion that can be *defined* internally. We strongly suspect that this is not the case for even the weak version propositional truncation, but we lack a formal proof.

It turns out to be interesting to consider the assumption that every type has a constant endofunction. The empty type has a trivial such endofunction, and so does a type of which we know an explicit inhabitant; however, from the assumption that a type has a weakly constant endofunction, we have no way of knowing in which case we are. In a minimalistic theory, we do not think that this assumption implies excluded middle. However, it implies that all equalities are decidable, i.e. a strong version of excluded middle holds for equalities.

Finally, we show that the judgmental computation rule of propositional truncation, if it is assumed, does have some interesting consequences for the theory. The most counter-intuitive observation is that the projection map $|-| : A \to \|A\|$ does at least meta-theoretically not hide any information. For example, it is possible to write down a term $\mathsf{myst}_{\mathbb{N}}$ such that $\mathsf{myst}_{\mathbb{N}} \circ |-| : \mathbb{N} \to \mathbb{N}$ is the identity function, which seems to show that the identity function factors through $\|\mathbb{N}\|$. Of course, this is neither possible (as $\|\mathbb{N}\|$ is equivalent to the unit type) nor actually the case.

Some parts of the Sections 3, 4, 6, 7 and 8 of this article have been published in our previous conference paper [14].

Formalization. We have formalized [15] all of our results in the dependently typed programming language and proof assistant *Agda* [21]. It is available in browser-viewable format and as plain source code on the first-named author's academic homepage. All proofs type check with the current Agda version 2.3.3 and we expect them to work with future Agda releases.

Our results have, like many results in Homotopy Type Theory, the property that they can be formalized in a very readable way, understandable even for readers who do not have any experience with the specific proof assistant, or formalized proofs in general. We have tried our best and would like to encourage the reader to give it a try.

Contents. In Section 2, we specify the type theory that we work in, a standard form Martin-Löf Type Theory. We also state basic definitions, but we try to use standard notation and we hope that all notions are as intuitive as possible. We then revisit Hedberg's Theorem in Section 3 and formulate several generalizations. Next, we move on to explore weakly constant constant functions between general types. We show that a constant endofunction has a propositional type of fixed points and factorizes through $\|-\|$ in Section 4. We

discuss why the factorization can not always be done for functions between different types in Section 5, together with some cases in which it is actually possible. In Section 6, we prove that, if every type has a constant endofunction, then all equalities are decidable. Section 7 is devoted to *populatedness*, a new definable notion of anonymous existence in type theory, based on our previous observations of constant endofunctions. We examine the differences between inhabitance, populatedness, propositional truncation and double negation, all of which are notions of existence, carefully in Section 8. Finally, Section 9 discusses consequences of the judgmental computation rule of propositional truncation, and Section 10 presents a summary and questions which we do not know the answer of.

2. Preliminaries

Our setting is a standard version of intensional Martin-Löf Type Theory (MLTT) with type universes that have coproducts, dependent sums, dependent products and identity types. We give a very rough specification of these constructions below. For a rigorous treatment, we refer to our main reference [27, Appendix A.1 or A.2]. We use standard notation whenever it is available. If it improves the readability, we allow ourselves to implicitely uncurry functions and write $f(x,y)$ instead of $f(x)(y)$ or $f\,x\,y$.

Type Universes. MLTT usually comes equipped with a hierarchy $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$ of universes, where $\mathcal{U}_{n+1}$ is the type of $\mathcal{U}_n$. Without very few exceptions, we only need one universe $\mathcal{U}$ and therefore omit the index. $\mathcal{U}$ can be understood as a generic universe or, for simplicity, as the lowest universe $\mathcal{U}_0$. If we say that X is a type, we mean $X : \mathcal{U}$, possibly in some context.

Coproducts. If X and Y are types, then so is $X + Y$. If we have $x : X$ or $y : Y$, we get $\mathsf{inl}\, x : X + Y$ or $\mathsf{inr}\, y : X + Y$, respectively. To prove a statement for all elements in $X + Y$, it is enough to consider those that are of one of these two forms.

Dependent Pairs. If X is a type and $Y : X \to \mathcal{U}$ a family of types, indexed over X, then $\Sigma_X\, Y$ is the corresponding *dependent pair type*, sometimes called a *dependent sum* or just Σ-*type*. For $x : X$ and $y : Y(x)$, we have $(x, y) : \Sigma_X\, Y$, and to eliminate out of $\Sigma_X\, Y$, it is enough to consider elements of this form. We prefer to write $\Sigma_{x:X}\, Y(x)$ instead of $\Sigma_X\, Y$, hoping to increase readability. Instead of $\Sigma_{x_1:X}\, \Sigma_{x_2:X}\, Y(x_1, x_2)$, we write $\Sigma_{x_1,x_2:X}\, Y(x_1, x_2)$. In the special case that Y does not depend on X, it is standard to write $X \times Y$.

Dependent Functions. Given $X : \mathcal{U}$ and $Y : X \to \mathcal{U}$ as before, we have the type $\Pi_X\, Y$, called the *dependent functions type* or Π-type. It is sometimes also referred to as the *dependent product type*, although that notion can be confusing as it would fit for Σ-types as well. If, for any given $x : X$, the term t is an element in $Y(x)$, we have $\lambda x.t : \Pi_X\, Y$. Synonymously to $\Pi_X\, Y$, we write $\Pi_{x:X}\, Y(x)$ or $\forall (x : X).\, Y(x)$. If the type X is not relevant or can be inferred, we shorten the latter notation to $\forall x.\, Y(x)$. Instead of $\Pi_{x_1:X}\, \Pi_{x_2:X}\, Y(x_1, x_2)$, we write $\Pi_{x_1,x_2:X}\, B(x_1, x_2)$ or $\forall (x_1, x_2 : X).\, B(x_1, x_2)$.

Identity Types. Given a type X with elements $x, y : X$, we have the *identity type* or the type of *equalities*, written $x =_X y$. An inhabitant $p : x =_X y$ shows that x and y are *propositionally equal*. p is called an *equality proof* or, having the interpretation of a type as a space in mind, a *path* from x to y. Similarly, $x =_X y$ is called a *path space*. The only introduction rule for the identity types is that, for any $x : X$, there is $\mathsf{refl}_x : x =_X x$. The elimination rule (called J) says that, if $P : (\Sigma_{x,y:X}\, x =_X y) \to \mathcal{U}$ is a type family, it suffices to construct an inhabitant of $\Pi_{x:X}\, P(x, x, \mathsf{refl}_x)$ in order to get an element of $P(p)$ for any $p : \Sigma_{x,y:X}\, x =_X y$. We do explicitly not assume other elimination rules such as *Streicher's*

K or *uniqueness of identity proofs (UIP)* [25]. If the common type of x, y can be inferred or is unimportant, we write $x = y$ instead of $x =_X y$.

In contrast to propositional equality, *definitional* (also called *judgmental*) equality is a meta-level concept. It refers to two terms, rather than two (hypothetical) elements, with the same β (and, sometimes, η in a restricted sense) normal form. Recently, it has become standard to use the symbol $\equiv$ for judgmental equality in order to use $=$ solely for propositional equality [27]. Note that the introduction rule of the latter says precisely that two judgmentally equal terms are, as elements of some type, propositionally equal. For definitions, we use the notation $:\equiv$.

Applying the eliminator J for propositional equality is also referred to as *path induction* [27]. A variant of J that is sometimes more useful is due to Paulin-Mohring [23]: given a point $x : X$ and a type family $P : (\Sigma_{y:X}\, x =_X y) \to \mathcal{U}$, it is enough to construct an inhabitant of $P(x, \mathsf{refl}_x)$ in order to construct an inhabitant of $P(y, q)$ for any pair (y, q). This elimination principle, called *based path induction*, is equivalent to J.

As a basic example, we show that propositional equality satisfies the *groupoid laws* [10], where reflexivity plays the role of identities. If we have $p : x =_X y$ and $q : y =_X z$, we can construct a path $p \centerdot q : x =_X z$ (the *composition* of p and q): by based path induction, it is enough to do this under the assumption that $(z, q) : \Sigma_{z:X}\, y =_X z$ is (y, refl_y). But in that case, the composition $p \centerdot q$ is given by p. Similarly, for $p : x =_X y$, there is $p^{-1} : y =_X x$. It is easy to see (again by path induction) that the types $p \centerdot \mathsf{refl}_y =_X p$ and $\mathsf{refl}_x \centerdot p =_X p$ as well as $p \centerdot p^{-1} =_X \mathsf{refl}_x$ are inhabited, and similarly, so are all the other types that are required to give a type the structure of a groupoid.

An important special case of the eliminator J is *substitution* or *transportation*: if $P : X \to \mathcal{U}$ is a family of types and $x, y : X$ are two elements (or points) that are equal by $p : x =_X y$, then an element of $e : P(x)$ can be *transported along the path* p to get an element of $P(y)$, written

$$p_*(e) : P(y). \tag{2.1}$$

Another useful function, similarly easily derived from J, is the following: if $f : X \to Y$ is a function and $p : x =_X y$ a path, we get an inhabitant of $f(x) = f(y)$ in Y,

$$\mathsf{ap}_f p : f(x) = f(y). \tag{2.2}$$

Note that we omit the arguments x and y in the notation of ap_f.

Identity types also enable us to talk about isomorphism, or (better) *equivalence*, of types. We say that X and Y are equivalent, written $X \simeq Y$, if there are functions in both directions which are the inverses of each other,

$$f : X \to Y \tag{2.3}$$

$$g : Y \to X \tag{2.4}$$

$$p : \forall(x : X).\, g(f(x)) =_X x \tag{2.5}$$

$$q : \forall(y : Y).\, f(g(y)) =_Y y. \tag{2.6}$$

Technically, (f, g, p, q) only constitute what is usually called a *type isomorphism*, but from any such isomorphism, an equivalence (in the sense of Homotopy Type Theory) can be constructed; and the only difference is that an equivalence requires a certain coherence between the components p and q, which will not be important for us. In this sense, we do not distinguish between isomorphims and equivalences, and only choose the latter terminology on principle. For details, we refer to [27, Chapter 4]. A small caveat is required: We call

types *logically equivalent*, written $X \longleftrightarrow Y$, if there are functions in both directions (that is, we only have the components (2.3) and (2.4)).

Equivalent types share all internalizable properties. In fact, Voevodsky's *Univalence Axiom* (e.g. [27], [29]) implies that equivalent types are propositionally equal. For the biggest part of our article, we do not need to assume the Univalence Axiom; however, it will play some role in Section 9.

We sometimes use other additional principles (namely *function extensionality* and *propositional truncation*, as introduced later). However, we treat them as assumptions rather than parts of the core theory and state clearly in which cases they are used.

In order to support the presentation from the next section on, we define a couple of notions. Our hope is that all of these are as intuitive as possible, if not already known. The only notions that are nonstandard or possibly ambiguous are *weak constancy*, meaning that for any pair of possible arguments for a function its values will be equal, *collapsible*, meaning that a type has such a weakly constant endofunction, and *path-collapsible*, saying that every path space over the type is collapsible.

Definition 2.1. We say that a type X is a *proposition* or *propositional* if all its inhabitants are equal:

$$\mathsf{isProp}\, X :\equiv \forall(x, y : X).\, x = y. \tag{2.7}$$

It is a well-known fact that the path spaces of a propositional type are not only inhabited but also propositional themselves. This stronger property is called *contractible*,

$$\mathsf{isContr}\, X :\equiv X \times \mathsf{isProp}\, X. \tag{2.8}$$

It is easy to see that any contractible type is equivalent to the unit type. An important class of contractible types are the *singletons*, or *path-to/path-from* types: for any $a_0 : A$, the type

$$\Sigma_{a:A}\, a_0 = a \tag{2.9}$$

is contractible, as any inhabitant is by based path induction easily seen to be equal to $(a_0, \mathsf{refl}_{a_0})$.

Further, X satisfies *UIP*, or is a *set*, if its path spaces are all propositional:

$$\mathsf{isSet}\, X :\equiv \forall(x, y : X).\ \mathsf{isProp}(x = y). \tag{2.10}$$

The properties of being contractible, propositional or a set are all propositional themselves, which the following properties are not.

X is *decidable* if it is either inhabited or empty,

$$\mathsf{decidable} X :\equiv X + \neg X. \tag{2.11}$$

We therefore say that X has *decidable equality* if the equality type of any two inhabitants of X is decidable:

$$\mathsf{discrete} X :\equiv \forall(x, y : X).\, \mathsf{decidable}(x = y). \tag{2.12}$$

Based on the terminology in [20], we also call a type with decidable equality *discrete*.

A function (synonymously, map) $f : X \to Z$ is *weakly constant*, or *1-constant*, if it maps any two elements to the same inhabitant of Y:

$$\mathsf{const}\, f :\equiv \forall(x, y : X).\, f(x) = f(y). \tag{2.13}$$

As weak (or 1-) constancy is the only notion of constancy that we consider in this article (if we ignore factorizability through $\|-\|$), we call such a function f just *constant* for simplicity. However, note that this notion is indeed very weak as soon as we consider functions into

types that are not sets, as we will see later. We call a type X *collapsible* if it has a weakly constant endomap:

$$\mathsf{coll}\, X :\equiv \Sigma_{f:X\to X}\ \mathsf{const}\, f. \tag{2.14}$$

Finally, X is called *path-collapsible* if any two points x, y of X have a collapsible path space:

$$\mathsf{pathColl}\, X :\equiv \forall(x, y : X).\ \mathsf{coll}\,(x = y). \tag{2.15}$$

For some statements, but only if clearly indicated, we use *function extensionality*. This principle says that two functions f, g of the same type are equal as soon as they are pointwise equal:

$$(\forall x.\, f(x) = g(x)) \to f = g. \tag{2.16}$$

An important equivalent formulation due to Voevodsky [30] is that the type of propositions is closed under Π ; more precisely,

$$(\forall x.\ \mathsf{isProp}\ Y(x)) \ \to\ \mathsf{isProp}\ \Pi_X\, Y. \tag{2.17}$$

In the case of non-dependent functions, this means that $X \to Y$ is propositional as soon as Y is.

A principle that we do not assume, but which will appear in some of our discussions, are the *law of excluded middle* in the form for propositions and in the form for general types [27, Chapter 3.4]. The first says that every proposition is decidable, while the second implies the same without the restriction to propositions.

$$\mathsf{LEM} :\equiv \forall(P : \mathcal{U}).\,(\mathsf{isProp}\, P) \to P + \neg P \tag{2.18}$$

$$\mathsf{LEM}_\infty :\equiv \forall(X : \mathcal{U}).\, X + \neg X. \tag{2.19}$$

Note that LEM_∞ can be considered the natural formulation under the *propositions as types* view. However, it includes a very strong form of choice which is inconsistent with the *Univalence Axiom* that is assumed in Homotopy Type Theory. Therefore, we consider LEM the "correct" formulation in our context.

3. Hedberg's Theorem

Before discussing possible generalizations, we want to state Hedberg's Theorem.

Theorem 3.1 (Hedberg [9])**.** *Every discrete type satisfies UIP,*

$$\mathsf{discrete}X \to \mathsf{isSet}\, X. \tag{3.1}$$

We shortly give Hedberg's original proof, consisting of two steps.

Lemma 3.2. *If a type has decidable equality, it is path-collapsible:*

$$\mathsf{discrete}X \to \mathsf{pathColl}\, X. \tag{3.2}$$

Proof. Given inhabitants x and y of X, we get by assumption either an inhabitant of $x = y$ or an inhabitant of $\neg(x = y)$. In the first case, we construct the required constant function $(x = y) \to (x = y)$ by mapping everything to this given path. In the second case, we have a proof of $\neg(x = y)$, and the canonical function is constant automatically. □

Lemma 3.3. *If a type is path-collapsible, it satisfies UIP:*

$$\mathsf{pathColl}\, X \to \mathsf{isSet}\, X. \tag{3.3}$$

Proof. Assume f is a parametrized constant endofunction on the path spaces, meaning that, for any $x, y : X$, we have a constant function $f_{x,y} : x = y \to x = y$. Let p be a path from x to y. We claim that

$$p = (f_{x,x}(\mathsf{refl}_x))^{-1} \cdot f_{x,y}(p). \tag{3.4}$$

By path induction, we only have to give a proof if the triple (x, y, p) is in fact (x, x, refl_x), which is one of the groupoid laws that propositional equality satisfies. Using the fact f is constant on every path space, the right-hand side of the above equality is independent of p, and in particular, equal to any other path of the same type. □

Hedberg's proof [9] is just the concatenation of the two lemmata. A slightly more direct proof can be found in the HoTT Coq repository [26] and in a post by the first named author on the HoTT blog [12].

Lemma 3.2 uses the rather strong assumption of decidable equality. In contrast, the assumption of Lemma 3.3 is equivalent its conclusion, so that there is no space for a strengthening. We include a proof of this simple claim in Theorem 3.10 below and concentrate on weakening the assumption of Lemma 3.3. Let us first introduce the notions of *stability* and *separatedness*.

Definition 3.4. For any type X, define

$$\mathsf{stable}\, X :\equiv \neg\neg X \to X, \tag{3.5}$$

$$\mathsf{separated}\, X :\equiv \forall(x, y : X).\ \mathsf{stable}(x = y). \tag{3.6}$$

We can see $\mathsf{stable}\, X$ as a *classical* condition, similar to $\mathsf{decidable} X \equiv X + \neg X$, but strictly weaker. Indeed, we get a first strengthening of Hedberg's Theorem as follows:

Lemma 3.5 ([27, Corollary 7.2.3])**.** *Any separated type is a set if function extensionality holds,*

$$\mathsf{separated}\, X \to \mathsf{isSet}\, X. \tag{3.7}$$

Proof. There is, for any $x, y : X$, a canonical map $(x = y) \to \neg\neg(x = y)$. Composing this map with the proof that X is separated yields an endofunction on the path spaces. With function extensionality, the first map has a propositional codomain, implying that the endofunction is constant and thereby fulfilling the requirements of Lemma 3.3. □

We remark that full function extensionality is actually not needed here. Instead, a weaker version that only works with the empty type is sufficient. Similar statements hold true for all further applications of extensionality in this paper. Details can be found in the Agda file [1].

In a constructive setting, the question how to express that "there exists something" in a type X is very subtle. One possibility is to ask for an inhabitant of X, but in many cases, this is too strong to be fulfilled. A second possibility, which corresponds to our above definition of *separated*, is to ask for a proof of $\neg\neg X$. Then again, this is very weak, and often too weak, as one can in general only prove negative statements from double-negated assumptions.

This fact has inspired the introduction of *squash types* (the Nuprl book [6]), and similar, *bracket types* (Awodey and Bauer [4]). These lie in between of the two extremes mentioned above. In our intensional setting, we talk of *propositional truncations*, or -1*-truncations* [27,

Chapter 3.7]. For any type X, we postulate that there is a type $\|X\|$ that is a *proposition*, representing the statement that X is inhabited. The rules are that if we have a proof of X, we can, of course, get a proof of $\|X\|$, and from $\|X\|$, we can conclude the same statements as we can conclude from X, but only if the actual representative of X does not matter:

Definition 3.6. We say that a type theory has *weak propositional truncations* if, for every type X, we have a type $\|X\| : \mathcal{U}$ which satisfies the following properties:

(1) $|-| : X \to \|X\|$
(2) $\mathsf{h_{tr}} : \mathsf{isProp}(\|X\|)$
(3) $\mathsf{rec_{tr}} : \forall(P : \mathcal{U}).\ \mathsf{isProp}\, P \to (X \to P) \to \|X\| \to P$.

Note that this amounts to saying that the operator $\|-\|$ is left adjoint to the inclusion of the subcategory of propositions into the category of all types. Therefore, it can be seen as the *propositional reflection*. For $x, y : \|X\|$, we will write $\mathsf{h_{tr}}_{x,y}$ for the proof of $x =_{\|X\|} y$ that we get from $\mathsf{h_{tr}}$.

Adopting the terminology of [27, Chapter 3.10], we say that X is *merely* inhabited if $\|X\|$ is inhabited. We may also say that X *merely* holds. However, we try to always be precise by giving the formal type expression to support the informal statement.

The *non-dependent eliminator* (or *recursion principle*, cf. [27, Chapter 5.1]) $\mathsf{rec_{tr}}$ implies the *dependent* one (the *induction principle*):

Lemma 3.7. *The propositional truncation admits the following induction principle: Given a type X, a family $P : \|X\| \to \mathcal{U}$ with a proof $h : \forall(z : \|X\|).\ \mathsf{isProp}(P(z))$, a term $f : \forall(x : X).\, P(|x|)$ gives rise to an inhabitant of $\forall(z : \|X\|).\, P(z)$.*

Proof. We have a map $j : X \to \Sigma_{z:\|X\|} P(z)$ by $\lambda x.(|x|, f(x))$. Observe that the codomain of j is a proposition, combining the fact that $\|X\|$ is one with h. Therefore, we get $\|X\| \to \Sigma_{z:\|X\|} P(z)$, and this is sufficient, using that $y =_{\|X\|} z$ for any $y, z : \|X\|$. □

In analogy to the notation $\mathsf{rec_{tr}}$, we may write $\mathsf{ind_{tr}}$ for the term witnessing this induction principle. However, most of our further developments will not benefit significantly, or not a all, from the induction principle, and will be proved with $\mathsf{rec_{tr}}$.

In contrast to other sources [27] we do *not* assume the judgmental β-rule

$$\mathsf{rec_{tr}}(P, h, f, |x|) \ \equiv_\beta \ f(x) \tag{3.8}$$

as it is simply not necessary for our results and we do not want to make the theory stronger than required. We do think that 3.8 is often useful, but we also think it is interesting to make clear in which sense 3.8 makes the theory actually stronger, rather than more convenient. We will discuss this in Section 9. A practical advantage of not assuming 3.8 is that the truncation can be implemented in existing proof assistants more easily. Of course, the β-rule holds propositionally as both sides of the equation inhabit the same proposition.

Note that $\|-\|$ is *functorial* in the sense that any function $f : X \to Y$ gives rise to a function $\|f\| : \|X\| \to \|Y\|$, even though the proof of $\|g \circ f\| = \|g\| \circ \|f\|$ requires function extensionality.

There is a type expression that is equivalent to propositional truncation:

Theorem 3.8. *For any given $X : \mathcal{U}$, we have*

$$\|X\| \ \longleftrightarrow \ \forall(P : \mathcal{U}).\ \mathsf{isProp}\, P \to (X \to P) \to P. \tag{3.9}$$

A potential problem with the expression on the right-hand side is that it is not living in universe $\mathcal{U}$. This size issue is the only thing that keeps us from using it as the definition for $\|X\|$. All other properties of the above Definition 3.6 are satisfied, at least under the assumption of function extensionality. Voevodsky [30] uses *resizing rules* to get rid of the problem.

Proof. The direction "$\to$" of the statement is not more than a rearrangement of the assumptions of property (3). For the other direction, we only need to instantiate P with $\|X\|$ and observe that the properties (1) and (2) in the definition of $\|X\|$ are exactly what is needed. □

With this definition at hand, we can provide an even stronger variant of Hedberg's Theorem. Completely analogously to the notions of stability and separatedness, we define *h-stable* and *h-separated*:

Definition 3.9. For any type X, define

$$\mathsf{hStable}\, X :\equiv \|X\| \to X, \tag{3.10}$$

$$\mathsf{hSeparated}\, X :\equiv \forall(x, y : X).\ \mathsf{hStable}(x = y). \tag{3.11}$$

We observe that $\mathsf{hSeparated}\, X$ is a strictly weaker condition than $\mathsf{separated}\, X$. Not only can we conclude $\mathsf{isSet}\, X$ from $\mathsf{hSeparated}\, X$, but the converse holds as well. In the following theorem, we also include the simple fact that path-collapsibility is equivalent to these statements.

Theorem 3.10. *For a type X in MLTT with propositional truncation, the following are equivalent:*

(1) X is a set
(2) X is path-collapsible
(3) X is h-separated.

Proof. "$3 \Rightarrow 1$" is just Lemma 3.3. "$1 \Rightarrow 3$" uses simply the the definition of the propositional truncation: given $x, y : X$, the fact that X is a set tells us exactly that $x = y$ is propositional, implying that we have a map $\|x = y\| \to (x = y)$. Concerning "$3 \Rightarrow 2$", it is enough to observe that the composition of $|-| : (x = y) \to \|x = y\|$ and the map $\|x = y\| \to (x = y)$, provided by the fact that X is h-separated, is a parametrized constant endofunction. □

We observe that using propositional truncation in some cases makes it unnecessary to appeal to functional extensionality: in Lemma 3.5, we have given a proof for the simple statement that separated types are sets in the context of function extensionality. This is not provable in plain MLTT. Let us now drop function extensionality and assume instead that propositional truncation is available. Every separated type is h-separated - more generally, we have

$$(\neg\neg X \to X) \to (\|X\| \to X) \tag{3.12}$$

for any type X -, and every h-separated space is a set. Notice that the mere availability of propositional truncation suffices to solve a gap that function extensionality would usually fill.

To conclude this part of the article, we want to mention that there is a slightly stronger version of the Hedberg's Theorem which applies to types where equality might only be decidable *locally*. In fact, nearly everything we stated or proved can be done locally, and thus made stronger. In the proof of Lemma 3.2, we have not made use of the fact that we were

dealing with path spaces at all: any decidable type trivially has a constant endofunction. Concerning Lemma 3.3, we observe:

Lemma 3.11 (Local form of Lemma 3.3)**.** *A locally path-collapsible type locally satisfies UIP:*

$$\forall x_0.\, (\forall y.\, \mathsf{coll}(x_0 = y)) \to \forall y.\, \mathsf{isProp}(x_0 = y). \tag{3.13}$$

Proof. The proof is identical to the one of Lemma 3.3, with the only difference that we need to apply based path induction instead of path induction. □

This enables us to prove the local variant of Hedberg's Theorem:

Theorem 3.12 ([22],[12]; Local form of Theorem 3.1)**.** *A locally discrete type is locally a set,*

$$\forall x_0.\, (\forall y.\, \mathsf{decidable}(x_0 = y)) \to \forall y.\, \mathsf{isProp}(x_0 = y). \tag{3.14}$$

□

In the same simple way, we immediately get that the assumption of local separatedness is sufficient.

Lemma 3.13 (Local form of Lemma 3.5)**.** *Under the assumption of function extensionality, a locally separated type locally is a set,*

$$\forall x_0.\, (\forall y.\, \mathsf{stable}(x_0 = y)) \to \forall y.\, \mathsf{isProp}(x_0 = y). \tag{3.15}$$

□

Similarly, the local forms of the characterizations of Theorem 3.10 are still equivalent.

Theorem 3.14 (Local form of Theorem 3.10)**.** *For a type X in MLTT with propositional truncation with a point $x_0 : X$, the following are equivalent:*

(1) for all $y : X$, the type $x_0 = y$ is propositional
(2) for all $y : X$, the type $x_0 = y$ is collapsible
(3) for all $y : X$, the type $x_0 = y$ is h-stable. □

Note that most of our arguments can be generalized to higher truncation levels [27, Chapter 7] in a reasonable and straightforward way. Details can be found in the first-named author's PhD thesis.[1]

4. Collapsibility implies H-Stability

If we unfold the definitions in the statements of Theorem 3.10, they all involve the path spaces over some type X:

(1) $\forall(x, y : X).\ \mathsf{isProp}(x = y)$
(2) $\forall(x, y : X).\ \mathsf{coll}\,(x = y)$
(3) $\forall(x, y : X).\ \mathsf{hStable}(x = y)$.

We have proved that these statements are logically equivalent. It is a natural question to ask whether the properties of path spaces are required. The possibilities that path spaces offer are very powerful and we have used them heavily. Indeed, if we formulate the above properties for an arbitrary type A instead of path types,

(1) $\mathsf{isProp}\, A$

[1] Not available yet.

(2) $\mathsf{coll}\, A$
(3) $\mathsf{hStable}\, A$,

we notice immediately that 1 is significantly and strictly stronger than the other two properties. 1 says that A has at most one inhabitant, 2 says that there is a constant endofunction on A, and 3 gives us a possibility to get an explicit inhabitant of A from the proposition that A has an anonymous inhabitant. A propositional type has the other two properties trivially, while the converse is not true. In fact, as soon as we know an inhabitant $a : A$, we can very easily construct proofs of 2 and 3, while it does not help at all with 1.

The implication $3 \Rightarrow 2$ is also simple: if we have $h : \|A\| \to A$, the composition $h \circ |-| : A \to A$ is constant, as for any $a, b : A$, we have $|a| = |b|$ and therefore $h(|a|) = h(|b|)$.

In summary, we have $1 \Rightarrow 3 \Rightarrow 2$ and we know that the first implication cannot be reversed. What is less clear is the reversibility of the second implication: If we have a constant endofunction on A, can we get a map $\|A\| \to A$? Put differently, what does it take to get out of $\|A\|$? Of course, a proof that A is h-stable is fine for that, but does a constant endomap on A also suffice? Surprisingly, the answer is positive, and there are interesting applications (Section 7). The main ingredient of our proof, and of much of the rest of the paper, is the following crucial lemma about fixed points:

Lemma 4.1 (Fixed Point Lemma)**.** *Given a constant endomap f on a type X, the type of fixed points is propositional, where this type is defined by*

$$\mathsf{fix}\, f :\equiv \Sigma_{x:X}\, x = f(x). \tag{4.1}$$

Before we can give the proof, we first need to formulate two observations. Both of them are simple on their own, but important insights for the Fixed Point Lemma. Let X and Y be two types.

Auxiliary Lemma 4.2 ([27, Theorem 2.11.3])**.** *Assume $h, k : X \to Y$ are two functions and $t : x = y$ as well as $p : h(x) = k(x)$ are paths. Then, transporting along t into p can be expressed as a composition of paths:*

$$t_*(p) = (\mathsf{ap}_h t)^{-1} \cdot p \cdot \mathsf{ap}_k t. \tag{4.2}$$

Proof. This is immediate by path induction on t. □

Even if the latter proof is trivial, the statement is essential. In the proof of Lemma 4.1, we need a special case, were x and y are the same. However, this special version cannot be proved directly. We consider the second observation a key insight for the Fixed Point Lemma:

Auxiliary Lemma 4.3. *If $f : X \to Y$ is constant and $x_1, x_2 : X$ are points, then $\mathsf{ap}_f : x_1 =_X x_2 \to f(x_1) =_Y f(x_2)$ is constant. In particular, ap_f maps every loop around x (that is, path from x to x) to $\mathsf{refl}_{f(x)}$.*

Proof. If c is the proof of $\mathsf{const}\, f$, then ap_f maps a path $p : x = y$ to $c(x, x)^{-1} \cdot c(x, y)$. This is easily seen to be correct for (x, x, refl_x), which is enough to apply path induction. As the expression is independent of p, the function ap_f is constant. The second part follows from the fact that ap_f maps refl_x to $\mathsf{refl}_{f(x)}$. □

With these lemmata at hand, we give a proof of the Fixed Point Lemma:

Proof of Lemma 4.1. Assume $f : X \to X$ is a function and $c : \mathsf{const}\, f$ is a proof that it is constant. For any two pairs (x, p) and $(x', p') : \mathsf{fix}\, f$, we need to construct a path connection them.

First, we simplify the situation by showing that we can assume that x and x' are the same: By composing $p : x = f\,x$ with $c(x, x') : f(x) = f(x')$ and $(p')^{-1} : f(x') = x'$, we get a path $p'' : x = x'$. By a standard lemma [27, Theorem 2.7.2], a path between two pairs corresponds to two paths: One path between the first components, and one between the second, where transporting along the first path is needed. We therefore now get that $(x, (p'')^{-1} \cdot p')$ and (x', p') are propositionally equal: p'' is a path between the first components, which makes the second component trivial. Write q for the term $(p'')^{-1} \cdot p'$.

We are now in the (nicer) situation that we have to construct a path between (x, p) and $(x, q) : \mathsf{fix}\, f$. Again, such a path can be constructed from two paths for the two components. Let us assume that we use some path $t : x = x$ for the first component. We then have to show that $t_*(p)$ equals q. In the situation with (x, p) and (x', p'), it might have been tempting to use p'' as a path between the first components, and that would correspond to choosing refl_x for t. However, one quickly convinces oneself that this cannot work in the general case.

By Auxiliary Lemma 4.2, with the identity for h and f for k, the first of the two terms, i. e. $t_*(p)$, corresponds to $t^{-1} \cdot p \cdot \mathsf{ap}_f t$. With Auxiliary Lemma 4.3, that term can be further simplified to $t^{-1} \cdot p$. What we have to prove is now just $t^{-1} \cdot p = q$, so let us just choose $p \cdot q^{-1}$ for t, thereby making it into a straightforward application of the standard lemmata. □

A more elegant but possibly less revealing proof of the Fixed Point Lemma was given by Christian Sattler:

Second Proof of Lemma 4.1 (Sattler). Given $f : X \to X$ and $c : \mathsf{const}\, f$ as before, assume $(x_0, p_0) : \mathsf{fix}\, f$. For any $x : X$, we have an equivalence of types,

$$f(x) = x \;\simeq\; f(x_0) = x, \tag{4.3}$$

given by precomposition with $c(x_0, x)$. Therefore, we also have the equivalence

$$\Sigma_{x:X}\, f(x) = x \;\simeq\; \Sigma_{x:X}\, f(x_0) = x. \tag{4.4}$$

The second of these types is (as a *singleton* or *path-from* type) contractible, while the first is just $\mathsf{fix}\, f$. This shows that any other inhabitant of $\mathsf{fix}\, f$ is indeed equal to (x_0, p_0). □

We will exploit Lemma 4.1 in different ways. For the following corollary note that, given an endomap f on X with constancy proof c, we have a canonical projection

$$\pi_1 : \mathsf{fix}\, f \to X \tag{4.5}$$

and a function

$$\epsilon : X \to \mathsf{fix}\, f \tag{4.6}$$

$$\epsilon(x) :\equiv (f(x)\,,\, c(x, f(x)))\,. \tag{4.7}$$

Corollary 4.4. *In basic MLTT, for a type X with a constant endofunction f, the type* $\mathsf{fix}\, f$ *is a proposition that is logically equivalent to X. In particular,* $\mathsf{fix}\, f$ *has all the properties that $\|X\|$ has (Definition 3.6). Therefore, the weak propositional truncation of collapsible types is actually* definable. *If $\|-\|$ is part of the theory, $\|X\|$ and* $\mathsf{fix}\, f$ *are equivalent.* □

We are now in the position to prove the statement that we have announced at the beginning of the section.

Theorem 4.5. *A type X is collapsible, i.e. has a constant endomap, if and only if it is h-stable in the sense that $\|X\| \to X$.*

Proof. As already mentioned in earlier, the "if"-part is simple: given $\|X\| \to X$, we just need to compose it with $|-| : X \to \|X\|$ to get a constant endomap. The other direction is an immediate consequence of Corollary 4.4. □

We want to add the remark that $\mathsf{coll}\, X$ is actually still more than required to get from $\|X\|$ to X. The following statement (together with the Theorem 4.5) shows that is is enough to have $f : X \to X$ which is *merely* constant:

Theorem 4.6. *For a type X, the following are logically equivalent:*

(1) X is collapsible
(2) X has an endofunction f with a proof $\|\mathsf{const}\, f\|$.

The first direction is trivial, but its reversibility is interesting. We do *not* think that $\|\mathsf{const}\, f\|$ implies $\mathsf{const}\, f$.

Proof of the nontrivial direction of Theorem 4.6. Assume f is an endofunction on X. From Lemma 4.1, we know that

$$\mathsf{const}\, f \to \mathsf{isProp}(\mathsf{fix}\, f). \tag{4.8}$$

Using the recursion principle with the fact that the statement $\mathsf{isProp}(\mathsf{fix}\, f)$ is a proposition itself yields

$$\|\mathsf{const}\, f\| \to \mathsf{isProp}(\mathsf{fix}\, f). \tag{4.9}$$

Previously, we have constructed a map

$$\mathsf{const}\, f \to \|X\| \to \mathsf{fix}\, f. \tag{4.10}$$

Let us write this implication as

$$\|X\| \to \mathsf{const}\, f \to \mathsf{fix}\, f. \tag{4.11}$$

This trivially implies

$$\|X\| \times \|\mathsf{const}\, f\| \to \mathsf{const}\, f \to \mathsf{fix}\, f. \tag{4.12}$$

We assume $\|X\| \times \|\mathsf{const}\, f\|$. From (4.9), we conclude that $\mathsf{fix}\, f$ is a proposition. Therefore, we may apply the recursion principle of the truncation and get

$$\|X\| \times \|\mathsf{const}\, f\| \to \|\mathsf{const}\, f\| \to \mathsf{fix}\, f, \tag{4.13}$$

which, of course, gives us

$$\|X\| \to \mathsf{fix}\, f \tag{4.14}$$

under the assumption 2 of the theorem. Composing $|-|$ with (4.14) and with the first projection, we get a constant function $g : X \to X$. □

Note that, in the above proof, we could have used the induction principle 3.7 instead of the "trick" of duplicating the assumption $\|\mathsf{const}\, f\|$.

Further, it seems to be impossible to show that the constructed function g is equal to f. On the other hand, it is easy to prove the truncated version of this statement:

$$\|\forall x.\, fx = gx\|. \tag{4.15}$$

The detailed proof can be found in our formalization [15].

5. Factorizing weakly constant Functions

In Theorem 4.5 we have seen that a constant function $f : X \to X$ implies that X is h-stable. On the other hand, what we have done is actually slightly more: the constructed map $\overline{f} : \|X\| \to X$ has the property that

$$\overline{f} \circ |-| : X \to X \tag{5.1}$$

is pointwise equal to f.

It seems a natural question to ask whether the fact that f is an endofunction is required: given a (weakly) constant function $f : X \to Y$, can it be factorized in this sense through $\|X\|$?

5.1. The Limitations of weak Constancy.

Let us start by giving a precise definition.

Definition 5.1. Given a function $f : X \to Y$ between two types, we say that f *factorizes* through a type Z if there are functions $f_1 : X \to Z$ and $f_2 : Z \to Y$ such that

$$\Pi_{x:X}\ f_2(f_1(x)) =_Y f(x). \tag{5.2}$$

In particular, we say that f factorizes through $\|X\|$ if there is a function $\overline{f} : \|X\| \to Y$ such that

$$\Pi_{x:X}\ \overline{f}(|x|) =_Y f(x). \tag{5.3}$$

We should indeed assume that a constant function $f : X \to Y$ factorizes through $\|X\|$ if we expect $\|X\|$ to be a *quotient* of X in the more "traditional" sense. Before the awareness of possibly non-propositional identity types was risen, the quotient X/R of X by a relation $R : X \times X \to \mathcal{U}$ was defined to have an eliminator that allows to construct a function $\overline{f} : (X/R) \to Y$ whenever a map $f : X \to Y$ with the property

$$\forall (x, y : X).\, R(x, y) \to f(x) = f(y) \tag{5.4}$$

is given [2]. If we want to view $\|X\|$ as X divided by the chaotic relation that relates each pair of elements of X, this elimination principle amounts exactly to the "lifting" of a constant function $X \to Y$ to a function $\|X\| \to Y$. This is indeed the case under the assumption of unique identity proofs as we will see later (Theorem 5.4).

However, the homotopical view suggests that it is unreasonable to expect such a lifting in the general case precisely because we have no way of knowing what happens on the (higher) path spaces. Consider the case that X is the coproduct of three propositions,

$X \equiv P + Q + R$. Let us write $\mathsf{in}_1 : P \to X, \mathsf{in}_2 : Q \to X, \mathsf{in}_3 : R \to X$ for the three embeddings. Assume that, for some function $f : X \to Y$, we have three "potential paths"

$$c_{12} : \Pi_{p:P}\, \Pi_{q:Q}\, f(\mathsf{in}_1 p) = f(\mathsf{in}_2 q), \tag{5.5}$$

$$c_{23} : \Pi_{q:Q}\, \Pi_{r:R}\, f(\mathsf{in}_2 q) = f(\mathsf{in}_3 r), \tag{5.6}$$

$$c_{13} : \Pi_{p:P}\, \Pi_{r:R}\, f(\mathsf{in}_1 p) = f(\mathsf{in}_3 r). \tag{5.7}$$

A priori, we do not know which of P, Q and R are inhabited so we do not know which of these paths actually exists. Exploiting that P, Q and R are propositional, it is straightforward to construct a proof that f is constant out of this data. Further, we get by the fact that $\|X\|$ is propositional the proofs

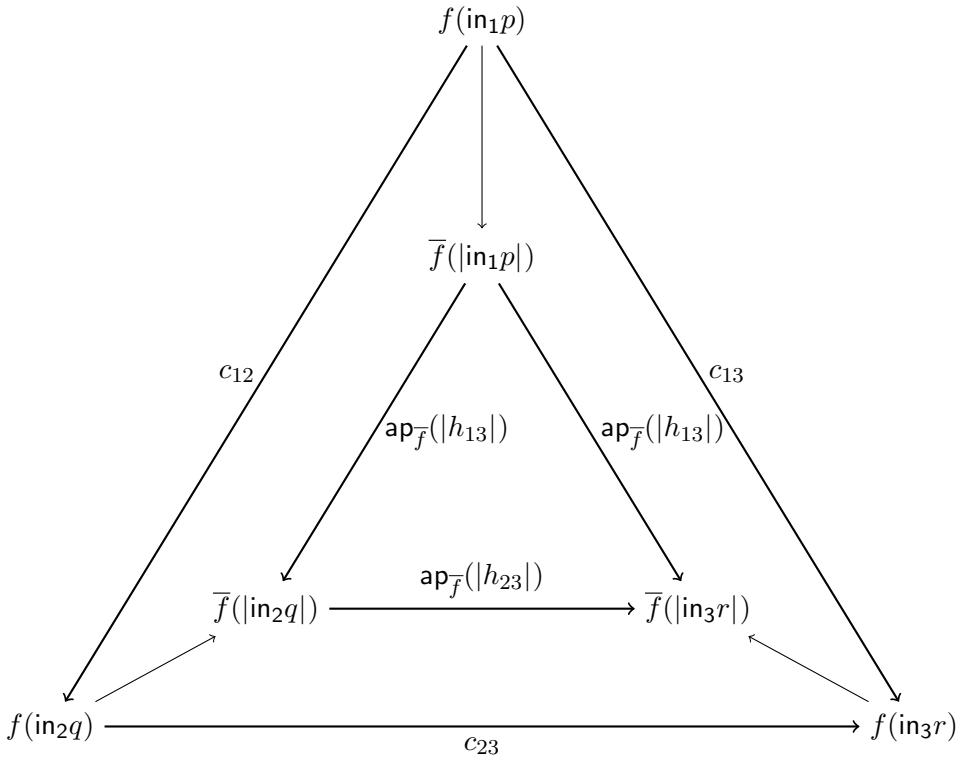
$$h_{12} : \Pi_{p:P}\, \Pi_{q:Q}\, |\mathsf{in}_1 p| = |\mathsf{in}_2 q|, \tag{5.8}$$

$$h_{23} : \Pi_{q:Q}\, \Pi_{r:R}\, |\mathsf{in}_2 q| = |\mathsf{in}_3 r|, \tag{5.9}$$

$$h_{13} : \Pi_{p:P}\, \Pi_{r:R}\, |\mathsf{in}_1 p| = |\mathsf{in}_3 r|. \tag{5.10}$$

Let us now assume that there is a way to factorize any generic constant function through the propositional truncation.

In our situation, we then get $\overline{f} : \|X\| \to Y$. If we are further given inhabitants $p : P$, $q : Q$ and $r : R$, let us observe that we have the following situation:



The arrows in the above diagram are the equality proofs, where we omit the arguments of c_{ij} and h_{ij}. The three unlabeled lines which "connect" the outer and the inner triangle are given by the fact that f and $f \circ \mathsf{ap}_{\_}$ are pointwise equal.

Note that $h_{12} \cdot h_{23} = h_{13}$ is automatically satisfied since $\|X\|$ is propositional. Looking at the smallest triangle in the above diagram we can conclude that it commutes due to the

usual functoriality of ap [27, Lemma 2.2.2], i.e. there is a proof of

$$\mathsf{ap}_{\overline{f}}(h_{12}) \cdot \mathsf{ap}_{\overline{f}}(h_{23}) = \mathsf{ap}_{\overline{f}}(h_{31}). \tag{5.11}$$

The large triangle will in general not commute as identity proofs are not necessarily unique.

If we only regard those parts of the diagram that do not mention the element r, we get a quadrangle in the top-left part. A similar observation holds for p and q so that we have three such quadrangles.

Let us go back one step. Assume that we are given the function f with the c_{ij} and (only) two points $p : P$ and $q : Q$. The path type $f(\mathsf{in}_1 p) = f(\mathsf{in}_2 q)$ is inhabited by $c_{12}(p, q)$. However, we should not expect to be able to construct an inhabitant that is not propositionally equal to this one. If we regard P and Q as two copies of the unit type, the only paths that we are able to construct are built out of refl and the c_{ij}. We further argue that the terms c_{23} and c_{13} can not be used for a generic R, as we can not know whether they actually provide any data (R could be empty). Therefore, if there is a way to construct the factorization of f, we expect it to not give us a "new" proof of $f(\mathsf{in}_1 p) = f(\mathsf{in}_2 q)$, i.e. we expect the quadrangle to commute, and the other quadrangles by analogous arguments. However, this contradicts the observation that the large triangle will in general not commute.

While this is not a rigorous argument, we hope that it provides some intuition. It seems that such a factorization would need to make some form of choice on the path spaces if the constancy proof does not satisfy certain coherence laws. In general, we cannot make such a non-canonical choice. We do not know whether the assumption that every constant function factors through the propositional truncation makes it possible to derive a clearer version of choice, such as LEM or the (propositional) axiom of choice [27, Chapter 3.8]. A meta-theoretic proof sketch that the factorization is not possible was described to us by Shulman [28].

5.2. **Factorization for Special Cases.** Even though we cannot factorize weakly constant functions in general, we can do it in some interesting special cases.

Constructing a function out of the propositional truncation of a type is somewhat tricky. A well-known [27, Chapter 3.9] strategy for defining a map $\|X\| \to Y$ is to construct a proposition P together with functions $X \to P$ and $P \to Y$. We have already implicitly done this in previous sections. We can make this method slightly more convenient to use if we observe that P does not need to be a proposition, but it only needs to be a proposition under the assumption that X is inhabited:

Principle 5.2. *Let X, Y be two types. Assume P is a type such that $P \to Y$. If X implies that P is contractible, then $\|X\|$ implies Y. In particular, if $f : X \to Y$ is a function that factorizes through P, then f factorizes through $\|X\|$.*

Let us shortly justify this principle. Assume that P has the assumed property. Utilizing that the statement that P is contractible is propositional itself, we see that $\|X\|$ is sufficient to conclude that P is a proposition. This allows us to prove $\|X\| \times P$ to be propositional. The map $P \to Y$ clearly gives rise to a map $\|X\| \times P \to Y$, and the map $X \to \|X\| \times P$ is given by $|-|$ and the fact that P is contractible under the assumption X. □

There are several situations in which this principle can be applied. The following theorem does not need it as it is mostly a restatement of our previous result from Section 4.

Theorem 5.3. *A weakly constant function $f : X \to Y$ factorizes through $\|X\|$ in any one of the following cases, of which the equivalent (3) and (4) generalize all others:*

(1) X is empty, i.e. $X \to \mathbf{0}$
(2) X is inhabited, i.e. $\mathbf{1} \to X$
(3) X is h-stable, i.e. $\|X\| \to X$
(4) X is collapsible, i.e. has a weakly constant endofunction
(5) we have any function $g : Y \to X$.

Proof. (1) and (2) both imply (3). Further, (5) implies (4) as the composition $g \circ f$ is a constant endofunction on X. The equivalence of (3) and (4) is Theorem 4.5. Thus, it is sufficient to prove the statement for (3), so assume $s : \|X\| \to X$. The required conclusion is then immediate as f is pointwise equal to the composition of $|-| : X \to \|X\|$ and $f \circ s$. □

Our next statement implies what we mentioned at the beginning of Section 5: under the assumption of unique identity proofs, the factorization is always possible.

Theorem 5.4. *Let X, Y be again two types and $f : X \to Y$ a constant function. If Y is a set, then f factorizes through $\|X\|$.*

Proof. We use Principle 5.2 and define

$$P :\equiv \Sigma_{y:Y} \|\Sigma_{x:X} f(x) =_Y y\|. \tag{5.12}$$

Given two elements (y_1, p_1) and (y_2, p_2) in P, we need to show that they are equal. Let us once more construct the equality via giving a pair of paths. For the second component, there is nothing to do as p_1 and p_2 live in propositional types. To show $y_1 =_Y y_2$, observe that this type is propositional as Y is a set and we may thus assume that we have inhabitants $(x_1, q_1) : \Sigma_{x_1:X} f(x_1) =_Y y_1$ and $(x_2, q_2) : \Sigma_{x_2:X} f(x_2) =_Y y_2$ instead of p_1 and p_2. But $f(x_1) = f(x_2)$ by constancy, and therefore $y_1 = y_2$. The maps $X \to P$ and $P \to Y$ are the obvious ones and the claim follows by Principle 5.2 (or rather the preceding comment, the strengthened version is not needed). □

It is not hard to see that, assuming function extensionality, the implication of Theorem 5.4 gives rise to an equivalence

$$(\Sigma_{f:X\to Y} \ \mathsf{const}\, f) \ \simeq \ (\|X\| \to Y)\,, \tag{5.13}$$

where we use in particular that $\mathsf{const}\, f$ is propositional under the given conditions. This is the simplest non-trivial special case of a general higher truncation elimination theorem that will be presented in [5].

Our last example of a special case in which the factorization can be done is more involved. However, it is worth the effort as it provides valuable intuition and an interesting application, as we will discuss below. The proof we give benefits hugely from a simplification by Sattler who showed to us how reasoning with type equivalences can be applied here.

Theorem 5.5. *Assume that function extensionality holds. If $f : X \to Y$ is constant and X is the coproduct of two propositions, then f factorizes through $\|X\|$.*

Proof. We make use of three basic properties. For each of them, assume that A is a type and $B : A \to \mathcal{U}$ a type family.

(1) *neutral contractible base space:* if A is propositional and $a_0 : A$, then $\Sigma_A B \simeq A(a_0)$. [27, Theorem 3.11.9 (ii)]
(2) *neutral contractible exponent:* if A is propositional and $a_0 : A$, we further have $\Pi_A B \simeq B(a_0)$.
(3) AC_∞ (*the* ∞*-"axiom of choice"*): if $C : (\Sigma_A B) \to \mathcal{U}$ is another type family, the equivalence

$$\left(\Pi_{a:A} \Sigma_{b:B(a)} Z(a,b)\right) \;\simeq\; \left(\Sigma_{g:\Pi_A B} \Pi_{a:A} C(a, g(a))\right) \tag{5.14}$$

holds. [27, Theorem 2.15.7]

Note that the third property is called ∞*-axiom of choice*, even though it is not an axiom but a derivable type equivalence.

Assume $X \equiv Q + R$, where Q and R are propositions. Define P to be the following Σ-type with four components:

$$\begin{aligned} P \;:\equiv\; & \Sigma\,(y : Y) \\ & \Sigma\,(s : \Pi_{q:Q}\; y = f(\mathsf{inl}\, q)) \\ & \Sigma\,(t : \Pi_{r:R}\; y = f(\mathsf{inr}\, r)) \\ & \quad \left(\Pi_{q:Q}\, \Pi_{r:R}\; s(q)^{-1} \cdot t(r) \;=\; c(\mathsf{inl}\, q\,,\, \mathsf{inr}\, r)\right) \end{aligned} \tag{5.15}$$

In order to apply Principle 5.2 we need to construct a function $P \to Y$ and a proof that X implies that P is contractible.

The function $P \to Y$ is, of course, given by a simple projection. For the other part, let a point of X be given. Without loss of generality, we assume that this inhabitant is $\mathsf{inl}\, q_0$ with $q_0 : Q$. As Christian Sattler has showed to us, constructing a chain of equivalences yields a nicer proof than the "naive" approach of finding a point that is equal to any other given inhabitant.

Let us first use the property of *neutral contractible exponents* (2): instead of quantifying over all elements of Q, it suffices to only consider q_0. Applying this twice shows that P is equivalent to the following type:

$$\begin{aligned} & \Sigma\,(y : Y) \\ & \Sigma\,(s : y = f(\mathsf{inl}\, q_0)) \\ & \Sigma\,(t : \Pi_{r:R}\; y = f(\mathsf{inr}\, r)) \\ & \quad \left(\Pi_{r:R}\; s^{-1} \cdot t(r) \;=\; c(\mathsf{inl}\, q_0\,,\, \mathsf{inr}\, r)\right). \end{aligned} \tag{5.16}$$

The first two components together match the definition of a *singleton* type, showing that this part is contractible with the canonical inhabitant $(f(\mathsf{inl}\, q_0), \mathsf{refl})$. Applying the principle of *neutral contractible base spaces* (1), the above type further simplifies to

$$\begin{aligned} & \Sigma\,(t : \Pi_{r:R}\; f(\mathsf{inl}\, q_0) = f(\mathsf{inr}\, r)) \\ & \quad \left(\Pi_{r:R}\; \mathsf{refl}^{-1} \cdot t(r) \;=\; c(\mathsf{inl}\, q_0\,,\, \mathsf{inr}\, r)\right). \end{aligned} \tag{5.17}$$

We apply the AC_∞ (3) in the direction from right to left and use that refl is neutral and self-inverse and neutral with respect to $\cdot$ to make a further transformation to

$$\begin{aligned} & \Pi_{r:R}\, \Sigma\,(t : f(\mathsf{inl}\, q_0) =_B f(\mathsf{inr}\, r)) \\ & \quad (t \;=\; c(\mathsf{inl}\, q_0\,,\, \mathsf{inr}\, r)). \end{aligned} \tag{5.18}$$

For any $r : R$, the dependent pair part is contractible as it is, once more, a *singleton* type, so that function extensionality implies the required result. □

Theorem 5.5 was inspired by a discussion on the *Homotopy Type Theory Mailing List* [28]. Shulman observed that, for two propositions Q and R, their *join* $Q * R$ [27, Chapter 6.8], defined as the (homotopy) pushout of the diagram $Q \xleftarrow{\pi_1} Q \times R \xrightarrow{\pi_2} R$, is equivalent to $\|Q + R\|$. This means that, in the presence of *higher inductive types* [27, Chapter 6], the type $\|Q + R\|$ has the (seemingly) stronger elimination rule of the join. The second named author then asked whether higher inductive types do really improve the elimination properties of $\|Q + R\|$ in this sense. This was discussed shortly before we could answer the question negatively with the result of Theorem 5.5: its statement about $\|Q + R\|$ corresponds exactly to the elimination property of $Q * R$. Thus, the join of two propositions already exists in a minimalistic setting that involves truncation but no other higher inductive types.

It is interesting to analyze how the factorizations constructed in Theorem 5.3, 5.4 and 5.5 bypass the difficulties discussed in Section 5.1. As explained above, performing such a factorization fails in general as the proof of weak constancy does usually not satisfy certain coherence properties and a factorization would thus require to make non-canonical choices of paths.

(1) If the codomain of a constant function is a set as in Theorem 5.4 this is resolved completely as parallel paths will automatically be equal.
(2) If, as in Theorem 5.5, the domain is the sum $Q+R$ of two propositions, the constancy proof is still not coherent in general. What we exploit is essentially that it can be replaced by a coherent one: given $f : Q + R \to Y$ and $c : \mathsf{const}\, f$, a coherent $c' : \mathsf{const}\, f$ can be constructed by mapping $(\mathsf{inl}\, q_1, \mathsf{inl}\, q_2)$ to the proof that is induced by the fact that Q is propositional, and similarly in the case of $(\mathsf{inr}\, r_1, \mathsf{inr}\, r_2)$. In the more interesting cases, $(\mathsf{inl}\, q, \mathsf{inr}\, r)$ is sent to $c(\mathsf{inl}\, q, \mathsf{inl}\, r)$, and $(\mathsf{inr}\, r, \mathsf{inl}\, q)$ to $c(\mathsf{inl}\, q, \mathsf{inr}\, r)^{-1}$.
(3) Consider a constant function $f : X \to Y$ together with any function $g : Y \to X$ as in Theorem 5.3. The function f does induce some form of asymmetry on the type Y. Usually, this asymmetry seems to be too weak to be useful, but the function g "sends it back" to the type X where it does allow us to make a choice of a point, namely the fixed point of the composition.

6. Global Collapsibility implies Decidable Equality

If X is some type, having a proof of $\|X\|$ is, intuitively, much weaker than a proof of X. While the latter consists of a concrete element of X, the first is given by an *anonymous* inhabitant of X. This is nothing more than the intention of the truncation: $\|X\|$ allows us to make the statement that "there exists something in X", without giving away a concrete element. It is therefore unreasonable to suppose that

$$\forall(X : \mathcal{U}).\, \|X\| \to X \tag{6.1}$$

can be proved, but it is interesting to consider what it would imply. Using Theorem 4.5, we can formulate the assumption in a weaker theory that does not have truncations:

Every type has a constant endomap. (6.2)

From a constructive type of view, this is an interesting statement. It clearly follows from LEM_∞: if we know an inhabitant of a type, we can immediately construct a constant endomap, and for the empty type, considering the identity function is sufficient. Thus, we

may understand *"Every type has a constant endomap"* as a form of excluded middle. It seems to use that every type is either empty or inhabited, but there is no way of knowing in which case we are.

If we assume (6.1), every type is h-stable. This holds in particular for path spaces and thus, every type is h-separated. By Theorem 3.10, every type is a set which is not a consistent assumption in Homotopy Type Theory. Already without the Univalence Axiom, (6.1) implies the Axiom of Choice [27, Chapter 3.8]. If we have univalence for propositions and set quotients, this allows us to use Diaconescu's proof of LEM ([7], cf. [27, Theorem 10.1.14]).

Let us consider a very minimalistic type theory without univalence, without function extensionality and without truncations. Under these conditions, 6.1 can not be expressed directly, so let us use the equivalent 6.2 instead: let us assume that every type has a constant endofunction (recall that this makes weak propositional truncation definable by Corollary 4.4, even though we will not use it directly). We do not think that LEM_∞ can be derived. However, what we can conclude is the ∞-version of excluded middle for all path spaces, i.e. that all types are discrete, see Lemma 6.1 and Theorem 6.2 below.

Lemma 6.1. *In basic MLTT without extensionality, without truncation, and even without a universe, let A be a type and $a_0, a_1 : A$ two points. If for all $x : A$ the type $(a_0 = x) + (a_1 = x)$ is collapsible, then $a_0 = a_1$ is decidable.*

Before giving the proof, we state an immediate corollary (which does involve a type universe):

Theorem 6.2. *If every type has a constant endofunction then every type has decidable equality,*

$$(\forall(X : \mathcal{U}).\ \mathsf{coll}\, X) \to \forall(X : \mathcal{U}).\, \mathsf{discrete} X. \tag{6.3}$$

□

Proof of Lemma 6.1. For (technical and conceptual) convenience, we regard the elements a_0, a_1 as a single map

$$a : \mathbf{2} \to A \tag{6.4}$$

and we use

$$E_x :\equiv \Sigma_{i:\mathbf{2}}\ a_i = x \tag{6.5}$$

in place of the type $(a_0 = x) + (a_1 = x)$. This is justified by the fact that the property of being collapsible is clearly closed under type equivalence. In a theory with propositional truncation, the *image* of a can be defined to be $\Sigma_{x:A} \|E_x\|$ [27, Definition 7.6.3]. By assumption, we have a family of constant endofunctions f_x on E_x, and by the discussion above, we can essentially regard the type

$$E :\equiv \Sigma_{x:A}\ \mathsf{fix}\, f_x, \tag{6.6}$$

which can be unfolded to

$$\Sigma_{x:A}\, \Sigma_{(i,p):E_x}\, f_x(i,p) = (i,p), \tag{6.7}$$

as the image of a. It is essentially the observation that we can define this image that allows us to mimic Diaconescu's argument. Clearly, a induces a map

$$r : \mathbf{2} \to E \tag{6.8}$$

$$r(i) :\equiv (a_i, \epsilon(i, \mathsf{refl}_{a_i})). \tag{6.9}$$

Using that the second component is an inhabitant of a proposition, we have

$$r(i) = r(j) \longleftrightarrow a_i = a_j. \tag{6.10}$$

The type E can be understood as the quotient of 2 by the equivalence relation $\sim$, given by $i \sim j \equiv a_i = a_j$. If E was the image of a in the ordinary sense [27, Definition 7.6.3], the axiom of choice would be necessary to find a section of r (cf. [27, Theorem 10.1.14]). In our situation, this section is given by a simple projection,

$$s : E \to \mathbf{2} \tag{6.11}$$

$$s(x, ((i, p), q)) :\equiv i. \tag{6.12}$$

It is easy to see that s is indeed a section of r in the sense of $\forall(e : E).\, r(s(e)) = e$. Given $(x, ((i, p), q)) : E$, applying first s, then r leads to $(a_i, \epsilon(i, \mathsf{refl}_{a_i}))$. Equality of these expressions is equality of the first components due to the propositional second component. But p is a proof of $a_i = x$. From that property, we can conclude that, for any $e_0, e_1 : E$,

$$e_0 = e_1 \longleftrightarrow s(e_0) = s(e_1). \tag{6.13}$$

Combining (6.10) and (6.13) yields

$$a_i = a_j \longleftrightarrow s(r(i)) = s(r(j)), \tag{6.14}$$

where the right-hand side is an equality in $\mathbf{2}$ and thus always decidable. In particular, $a_0 = a_1$ is hence decidable. □

7. Populatedness

In this section we discuss a notion of *anonymous existence*, similar, but weaker (see Section 8.2) than propositional truncation. It crucially depends on the Fixed Point Lemma 4.1. Let us start by discussing another perspective of what we have explained in Section 4.

Trivially, for any type X, we can prove the statement

$$\|X\| \to (\|X\| \to X) \to X. \tag{7.1}$$

By Lemma 4.5, this is equivalent to

$$\|X\| \to \mathsf{coll}\, X \to X, \tag{7.2}$$

and hence

$$\mathsf{coll}\, X \to \|X\| \to X, \tag{7.3}$$

which can be read as: If we have a constant endomap on X and we wish to get an inhabitant of X (or, equivalently, a fixed point of the endomap), then $\|X\|$ is sufficient to do so. We can additionally ask whether it is also necessary: can we replace the first assumption $\|X\|$ by something weaker? Looking at formula 7.1, it would be natural to conjecture that this is not the case, but it is. In this section, we discuss what it can be replaced by, and in Section 8.2, we give a proof that it is indeed weaker.

For answering the question what is needed to get from $\mathsf{hStable}\, A$ to A, let us define the following notion:

Definition 7.1 (populatedness)**.** For a given type X, we say that X is populated, written $\langle\langle X \rangle\rangle$, if every constant endomap on X has a fixed point:

$$\langle\langle X \rangle\rangle :\equiv \forall(f : X \to X).\ \mathsf{const}\, f \to \mathsf{fix}\, f, \tag{7.4}$$

where $\mathsf{fix}\, f$ is the type of fixed points, defined as in Lemma 4.1.

This definition allows us to comment on the question risen above. If $\langle\langle X \rangle\rangle$ is inhabited and X is collapsible, then X has an inhabitant, as such an inhabitant can be extracted from the type of fixed points by projection. Hence, $\langle\langle X \rangle\rangle$ instead of $\|X\|$ in 7.3 would be sufficient as well. Therefore,

$$\langle\langle X \rangle\rangle \to (\|X\| \to X) \to X. \tag{7.5}$$

At this point, we have to ask ourselves whether (7.5) is an improvement over (7.3). But indeed, we have the following property:

Theorem 7.2. *Any merely inhabited type is populated. That is, for any type X, we have*

$$\|X\| \to \langle\langle X \rangle\rangle. \tag{7.6}$$

Proof. Assume f is a constant endofunction on X. The claim follows directly from Corollary 4.4. □

In Section 8 we will see that $\langle\langle X \rangle\rangle$ is in fact strictly weaker than $\|X\|$. Note that from (7.5), Theorem 4.5 and Theorem 7.2 we immediately get the following:

Corollary 7.3. *For any type X, the following statements are logically equivalent:*

(1) X is collapsible,

$$\Sigma_{f:X\to X} \ \mathsf{const}\, f \tag{7.7}$$

(2) X is h-stable,

$$\|X\| \to X \tag{7.8}$$

(3) X is inhabited if it is populated,

$$\langle\langle X \rangle\rangle \to X. \tag{7.9}$$

In particular, if X is a proposition, (2) is always satisfied and we may conclude $\langle\langle X \rangle\rangle \to X$. □

In the presence of propositional truncation, we give an alternative characterization of populatedness. Recall that we indicate propositional truncation with the attribute *merely*.

Lemma 7.4. *In MLTT with propositional truncation, a type is populated if and only if the statement that it merely h-stable implies that it is merely inhabited, or equivalently, if and only if the statement that X is h-stable implies X. Formally, the following types are logically equivalent:*

(1) $\langle\langle X \rangle\rangle$

(2) $\big\|\|X\| \to X\big\| \to \|X\|$

(3) $(\|X\| \to X) \to X$.

Proof. We have already discussed (1) ⇒ (3) above (cf. 7.5). (3) ⇒ (2) follows from the functoriality of the truncation operator. For (2) ⇒ (1), assume we have a constant endofunction f on X. This implies $\|X\| \to X$, thus $\big\|\|X\| \to X\big\|$ and, by assumption, $\|X\|$. But $\|X\|$ is enough to construct a fixed point of f by Corollary 4.4. □

One more characterization of populatedness, and a strong parallel to mere inhabitance, is given by the following statement.

Theorem 7.5. *In MLTT, any given type X is populated if and only if any proposition that is logically equivalent to it holds,*

$$\langle\langle X\rangle\rangle \longleftrightarrow \forall(P : \mathcal{U}).\ \mathsf{isProp}\, P \to (P \to X) \to (X \to P) \to P. \tag{7.10}$$

Note that the only difference to the type expression in Theorem 3.8 is that we only quantify over *sub-propositions* of X, i. e. over those that satisfy $P \to X$, while we quantify over all propositions in the case of $\|X\|$. This again shows that $\|X\|$ is at least as strong as $\langle\langle X\rangle\rangle$.

Proof. Let us first prove the direction "$\to$". Assume a proposition P is given, together with functions $X \to P$ and $P \to X$. Composition of these gives us a constant endomap on X, exactly as in the proof of Theorem 3.10. But then $\langle\langle X\rangle\rangle$ makes sure that this constant endomap has a fixed point, which is (or allows us to extract) an inhabitant of X. Using $X \to P$ again, we get P.

For the direction "$\leftarrow$", assume we have a constant endomap f. We need to construct an inhabitant of $\mathsf{fix}\, f$. In the expression on the right-hand side, choose P to be $\mathsf{fix}\, f$, and everything follows from Corollary 4.4. □

The similarities between $\|X\|$ and $\langle\langle X\rangle\rangle$ do not stop here. The following statement, together with the direction "$\to$" of the theorem that we have just proved, is worth to be compared to the definition of $\|X\|$ (that is, Definition 3.6):

Theorem 7.6. *For any type X, the type $\langle\langle X\rangle\rangle$ has the following properties:*

(1) $X \to \langle\langle X\rangle\rangle$
(2) $\mathsf{isProp}(\langle\langle X\rangle\rangle)$ *(if function extensionality holds).*

Proof. The first point follows immediately from the (stronger) statement of Theorem 7.2. For the second, we use that $\mathsf{fix}\, f$ is a proposition (Lemma 4.1). By function extensionality, a (dependent) function type is propositional if the codomain is (cf. Section 2) and we are done. □

8. Taboos and Counter-Models

In this final section we look at the differences between the various notions of (anonymous) inhabitance we have encountered. We have, for any type X, the following chain of implications:

$$X \longrightarrow \|X\| \longrightarrow \langle\langle X\rangle\rangle \longrightarrow \neg\neg X. \tag{8.1}$$

The first implication is trivial and the second is given by Theorem 7.2. Maybe somewhat surprisingly, the last implication does not require function extensionality, as we do not need to prove that $\neg\neg X$ is propositional: to show

$$\langle\langle X\rangle\rangle \to \neg\neg X\ , \tag{8.2}$$

let us assume $f : \neg X$. But then, f can be composed with the unique function from the empty type into X, yielding a constant endomap on X, and obviously, this function cannot have a fixed point in the presence of f. Therefore, the assumption of $\langle\langle X\rangle\rangle$ would lead to a contradiction, as required.

Under the assumption of LEM, all implications of the chain 8.1 except the first can be reversed as it is easy to show

$$\forall(X : \mathcal{U}).\, (\|X\| + \neg\|X\|) \to \neg\neg X \to \|X\|. \tag{8.3}$$

Constructively, none of the implications of 8.1 should be reversible. To make that precise, we use *taboos*, showing that the provability of a statement would imply the provability of another, better understood statement, that is known to be not provable. As a second technique, we use models. In this section, we present the following discussions:

(1) We have already seen (cf. Section 6) that the assumption that the first implication can be reversed makes all equalities decidable (and thereby propositional), a homotopical taboo. As an alternative argument, if every type is h-stable, a form of choice that does not belong to type theory is implied. Moreover, we observe that $\|X\| \to X$ can be read as "the map $|-| : X \to \|X\|$ is a split epimorphism" (where the latter notion requires to be read with care), and we show that already the weaker assumption that it is an epimorphism implies that all types are sets.
(2) It would be wonderful if the second implication could be reversed, as this would imply that propositional truncation is definable in MLTT. However, this is logically equivalent to a certain weak version of the axiom of choice discussed below, which is not provable but holds under LEM.
(3) If the last implication can be reversed, LEM holds (a constructive taboo, which is not valid in recursive models). Together with the above observation, function extensionality implies that LEM holds if and only if $\neg\neg X \to \langle\langle X \rangle\rangle$ for all X.

8.1. **Inhabited and Merely Inhabited.** The question whether the first implication in the chain above can be reversed has already been analyzed in Section 6. This is impossible as long as equality is not globally decidable. Here, we wish to state another noteworthy consequence of the collapsibility of all types,

$$\forall(X : \mathcal{U}).\, \mathsf{coll}\, X, \tag{8.4}$$

which we know to hold if every type is h-stable (Theorem 4.5). We show that it implies a form of choice that does not pertain to intuitionistic type theory. In order to formulate and prove this, we need a few definitions.

We say that a relation $R : X \times X \to \mathcal{U}$ is *propositionally valued* if

$$\forall(x\, y : X).\, \mathsf{isProp}(R(x, y)). \tag{8.5}$$

The *R-image* of a point $x : X$ is

$$R_x :\equiv \Sigma_{y:X} R(x, y). \tag{8.6}$$

We say that R is *functional* if its point-images are all propositions:

$$\forall(x : X).\, \mathsf{isProp}\, R_x. \tag{8.7}$$

We say that two relations $R, S : X \times X \to \mathcal{U}$ *have the same domain* if

$$\forall(x : X).\, R_x \longleftrightarrow S_x, \tag{8.8}$$

and that S is a *subrelation* of R if

$$\forall(x\, y : X).\, S(x, y) \to R(x, y). \tag{8.9}$$

Theorem 8.1. *If all types are collapsible, then every binary relation has a functional, propositionally valued subrelation with the same domain.*

Proof. Assume that $R : X \times X \to U$ is given. For $x : X$, let $k_x : R_x \to R_x$ be the constant map given by the assumption 8.4 that all types are collapsible. Define further

$$S(x, y) :\equiv \Sigma_{a:R(x,y)}\, (y, a) = k_x(y, a). \tag{8.10}$$

Then S is a subrelation of R by construction. We observe that S_x is equivalent to $\mathsf{fix}(k_x)$ and therefore propositional (by Lemma 4.1), proving that S is functional. Together with Corollary 4.4, this further implies

$$R_x \longleftrightarrow \mathsf{fix}\, k_x \longleftrightarrow S_x, \tag{8.11}$$

showing that R and S have the same domain.

What remains to show is that $S(x, y)$ is always a proposition. Let $s, s' : S(x, y)$. As S_x is propositional we know $(y, s) =_{S_x} (y, s')$. It is a standard lemma ([27, Theorem 2.7.2]) that this type corresponds to a dependent pair type with components

$$p : y =_X y \tag{8.12}$$

$$q : p_*(s) =_{S(x,y)} s'. \tag{8.13}$$

In our case, as every type is a set, we have $p = \mathsf{refl}_y$, and q gives us the required proof of $s =_{S(x,y)} s'$. □

Instead of the logically equivalent formulation (8.4), let us now assume the original assumption that $|-|$ can be reversed, that is,

$$\forall (X : \mathcal{U}).\, \|X\| \to X. \tag{8.14}$$

Note that a map $h : \|X\| \to X$ is automatically a section of $|-| : X \to \|X\|$ in the sense of

$$\forall (z : \|X\|).\, |h(z)| = z \tag{8.15}$$

as any two inhabitants of $\|X\|$ are equal. Therefore, we may read (8.14) as:

$$\text{For any type } X, \text{ the map } |-| : X \to \|X\| \text{ is a } \textit{split epimorphism}. \tag{8.16}$$

We want to consider a weaker assumption, namely

$$\text{For any type } X, \text{ the map } |-| : X \to \|X\| \text{ is an } \textit{epimorphism}, \tag{8.17}$$

where we call $e : U \to V$ an *epimorphism* if, for any type W and any two functions $f, g : V \to W$, we have the implication

$$(\forall u.\, f(e\, u) = g(e\, u)) \to \forall v.\, f\, v = g\, v. \tag{8.18}$$

Of course, under function extensionality, e is an epimorphism if and only if, for all W, f, g, we have

$$f \circ e = g \circ e \to f = g. \tag{8.19}$$

A caveat is required. Our definition of *epimorphism* is the direct naive translation of the usual 1-categorical notion into type theory. However, the category of types and functions with propositional equality is not only an ordinary category, but rather an $(\omega, 1)$-category. The definition (8.18) makes sense in the sub-universe of sets [27, Chapter 10.1], where equalities are propositional. However, the property of being an epimorphism in our sense is not propositional and it could rightfully be argued that it might not be the "correct" definition in a context where not every type is a set, similarly as we argued that LEM_∞ is a

problematic version of the principle of excluded middle. Despite of this, we use the notion as we think that it helps providing an intuitive meaning to the plain type expression (8.18).

Lemma 8.2. *Let Y be a type. If the map $|-| : (y_1 = y_2) \to \|y_1 = y_2\|$ is an epimorphism for any points $y_1, y_2 : Y$, then Y is a set.*

Proof. Assume Y, y_1, y_2 are given. Define two functions

$$f, g : \|y_1 = y_2\| \to Y \tag{8.20}$$

by

$$f(q) :\equiv y_1, \tag{8.21}$$

$$g(q) :\equiv y_2, \tag{8.22}$$

that is, f and g are constant at y_1 and y_2, respectively.

With these concrete choices, our assumption (8.18) with $e \equiv |-|$ becomes

$$(y_1 = y_2 \to y_1 = y_2) \to (\|y_1 = y_2\| \to y_1 = y_2) \tag{8.23}$$

which, of course, implies

$$\|y_1 = y_2\| \to y_1 = y_2. \tag{8.24}$$

The statement of the lemma then follows from Theorem 3.10. □

In the following theorem, we include the main result of Section 6 to directly compare it with the second part:

Theorem 8.3. *In basic MLTT with weak propositional truncation,*

(1) if $|-| : X \to \|X\|$ is a split epimorphism for every X, then all types have decidable equality

(2) if $|-| : X \to \|X\|$ is an epimorphism for every X, then all types are sets.

Proof. The first part is a reformulation of Theorem 6.2, while the second part is a corollary of Lemma 8.2. □

8.2. **Merely Inhabited and Populated.** Assume that the second implication can be reversed, meaning that we have

$$\forall(X : \mathcal{U}).\, \langle\langle X \rangle\rangle \to \|X\|. \tag{8.25}$$

Repeated use of the Fixed Point Lemma leads to a couple of interesting logically equivalent statements.

In the previous subsection, we have discussed that we cannot show every type to be h-stable. However, a weaker version of this is provable:

Lemma 8.4. *For every type X, the statement that it is h-stable is populated,*

$$\langle\langle \|X\| \to X \rangle\rangle. \tag{8.26}$$

To demonstrate the different possibilities that the logically equivalent formulations of populatedness offer, we want to give more than one proof. The first one uses Definition 7.1:

First proof. Assume we are given a constant endofunction f on $\|X\| \to X$. We need to construct a fixed point of f, or correspondingly, any inhabitant of $\|X\| \to X$. By Theorem 4.5, a constant function $g : X \to X$ is enough for this. Given $x : X$, we may apply f on the function that is everywhere x, yielding an inhabitant of $\|X\| \to X$. Applying it on $|x|$ gives an element of X, and we define $g(x)$ to be this element. The proof that that f is constant immediately translates to a proof that g is constant. □

Alternatively, we can use the logically equivalent formulation of populatedness, proved in Theorem 7.5:

Second proof. Assume P is a proposition and we have a proof of

$$P \longleftrightarrow (\|X\| \to X). \tag{8.27}$$

We need to show P. The logical equivalence above immediately provides an inhabitant of $X \to P$, and, by the rules of the propositional truncation, therefore $\|X\| \to P$. Assume $\|X\|$. We get P, thus $\|X\| \to X$ with the above equivalence, and therefore X (using the assumed $\|X\|$ again). This shows $\|X\| \to X$, and consequently, P. □

If propositional truncation is available, we may also use that $\langle\!\langle - \rangle\!\rangle$ can be written in terms of $\|-\|$.

Third proof. Using Lemma 7.4, the statement that needs to be shown becomes

$$\Big\|\|\|X\| \to X\Big\| \to \|X\| \to X\Big\| \to \Big\|\|X\| \to X\Big\|. \tag{8.28}$$

By functoriality of $\|-\|$, it is enough to show

$$\Big(\Big\|\|X\| \to X\Big\| \to \|X\| \to X\Big) \to (\|X\| \to X)\,, \tag{8.29}$$

which is immediate. □

The assumption that populatedness and mere inhabitance are equivalent has a couple of "suspicious" consequences, as we want to show now.

Theorem 8.5. *In MLTT with weak propositional truncation, the following are logically equivalent:*

(1) every populated type is merely inhabited,

$$\forall(X : \mathcal{U}).\, \langle\!\langle X \rangle\!\rangle \to \|X\| \tag{8.30}$$

(2) every type is merely h-stable,

$$\forall(X : \mathcal{U}).\, \Big\|\|X\| \to X\Big\| \tag{8.31}$$

(3) every proposition is projective *in the following sense:*

$$\forall(P : \mathcal{U}).\ \mathsf{isProp}\, P \to \forall(Y : P \to \mathcal{U}).\, (\Pi_{p:P} \|Y(p)\|) \to \|\Pi_P\, Y\| \tag{8.32}$$

(note that this is the axiom of choice [27, Chapter 3.8] *for propositions, without the requirement that Y is a family of sets)*

(4) $\langle\!\langle - \rangle\!\rangle : \mathcal{U} \to \mathcal{U}$ is functorial *in the sense that*

$$\forall(X\, Y : \mathcal{U}).\, (X \to Y) \to (\langle\!\langle X \rangle\!\rangle \to \langle\!\langle Y \rangle\!\rangle), \tag{8.33}$$

where this naming is justified at least in the presence of function extensionality which implies that $\langle\!\langle X \rangle\!\rangle \to \langle\!\langle Y \rangle\!\rangle$ is propositional, ensuring $\langle\!\langle g \circ f \rangle\!\rangle = \langle\!\langle g \rangle\!\rangle \circ \langle\!\langle f \rangle\!\rangle$.

Proof. We show all the implications that we know nice arguments for, and those are sufficient (and even more than necessary) to prove the theorem.

The equivalence of the first two points follows easily from what we already know. (**1**) ⇒ (**2**) is an application of Lemma 8.4, while (**2**) ⇒ (**1**) follows easily from Lemma 7.4.

Regarding the equivalence of the first and the last point, (**1**) ⇒ (**4**) is also immediate by functoriality of $\|-\|$. On the other hand, if (4) holds, the map $|-|$ gives rise to a function $\langle\langle X\rangle\rangle \to \langle\langle \|X\| \rangle\rangle$, but for any propositional P, the types P and $\langle\langle P\rangle\rangle$ are equivalent.

Let us now show (1) ⇒ (3). Let P be some proposition and $Y : P \to \mathcal{U}$ some family of types. If we assume (1), it is then enough to prove

$$\Pi_{p:P} \|Y(p)\| \to \langle\langle \Pi_P Y \rangle\rangle. \tag{8.34}$$

By Lemma 7.4, it is enough to show

$$\Pi_{p:P} \|Y(p)\| \to (\|\Pi_P Y\| \to \Pi_P Y) \to \Pi_P Y. \tag{8.35}$$

Under several assumptions, one of them being that some $p_0 : P$ is given, we need to construct an inhabitant of $Y(p_0)$. Recall the principle of the *neutral contractible exponent* that we used in the proof of Theorem 5.5. Here, it allows us to replace $\Pi_P Y$ by $Y(p_0)$ and $\Pi_{p:P} \|Y(p)\|$ by $\|Y(p_0)\|$, and the type (8.35) becomes

$$\|Y(p_0)\| \to (\|Y(p_0)\| \to Y(p_0)) \to Y(p_0). \tag{8.36}$$

(3) ⇒ (2) can be seen easily by taking P to be $\|X\|$ and Y to be constantly X. □

Consider the third of the four statements in Theorem 8.5. When $Y(p)$ is a set with exactly two elements for every $p : P$, this amounts to *the world's simplest axiom of choice* [8], which fails in some toposes.

Corollary 8.6. *In MLTT with weak propositional truncation,* $\forall(X : \mathcal{U}). \langle\langle X\rangle\rangle \to \|X\|$ *is not derivable.* □

8.3. **Populated and Non-Empty.** If we can reverse the last implication of the chain, we have

$$\forall(X : \mathcal{U}). \neg\neg X \to \langle\langle X\rangle\rangle. \tag{8.37}$$

To show that this cannot be provable, we show that it would imply LEM, a constructive taboo.

Theorem 8.7. *With function extensionality, the following implication holds:*

$$(\forall(X : \mathcal{U}). \neg\neg X \to \langle\langle X\rangle\rangle) \;\to\; \mathsf{LEM}. \tag{8.38}$$

Proof. Assume P is a proposition. Then so is the type $P + \neg P$ (where we require function extensionality to show that $\neg P$ is a proposition). Hence, the identity function on $P + \neg P$ is constant.

On the other hand, it is straightforward to construct a proof of $\neg\neg(P + \neg P)$. By the assumption, this means that $P + \neg P$ is populated, i.e. every constant endomap on it has a fixed point. Therefore, we can construct a fixed point of the identity function, which is equivalent to proving $P + \neg P$. □

The other direction is standard. Thus, we have derived:

Corollary 8.8. *Under the assumption of function extensionality, all nonempty types are populated if and only if* LEM *holds.* □

9. Propositional Truncation with Judgmental Computation Rule

Propositional truncation is often defined to satisfy the judgmental computation rule [27, Chapter 3.7],

$$\mathsf{rec}_{\mathsf{tr}}(P, h, f, |x|) \equiv_\beta f(x) \tag{9.1}$$

for any function $f : X \to P$ where $x : X$ and P is propositional. In our discussion, we did not assume it to hold so far. We do certainly not want to argue that a theory without this judgmental equation is to be preferred, we simply did not need it. We agree with the very common view (see the introduction of [27, Chapter 6]) that judgmental computation rules are often advantageous, not only for truncations, but for *higher inductive types* [27, Chapter 6] in general. Without them, some expressions will need to involve a ridiculous amount of transporting, just to make them type check, and the "computation" will have to be done manually in order to simplify terms. If (9.1) is assumed, it suggests itself to also assume a judgmental computation rule for the induction principle, that is

$$\mathsf{ind}_{\mathsf{tr}}(P, h, f, |x|) \equiv_\beta f(x), \tag{9.2}$$

where $P : \|X\| \to \mathcal{U}$ might now be a type family and $f : \Pi_{z:\|X\|} P(z)$ is a dependent function rather than a simple function. An interesting aspect is that, unlike the propositional rule, it does not seem to follow from (9.1). In particular, the term constructed in Lemma 3.7 does *not* have the expected judgmental computation rule.

Having said this, the judgmental β-rules do have some other noteworthy consequences. Unlike the previous results, the statements in this part of our article do need the computation rules to hold judgmentally.n So far, all our lemmata and theorems have been internal to type theory. This is only partially the case for the results from this section, as any statement that some equality holds judgmentally is a meta-theoretic property. We thus can not implement such a statement as a type in a proof assistant such as Agda, but we can still use Agda to check our claims; for example, if

$$p : x = y \tag{9.3}$$

$$p :\equiv \mathsf{refl}_x \tag{9.4}$$

type checks, we may conclude that the equality does hold judgmentally.

9.1. **The Interval.** The interval $\mathbb{I}$ as a higher inductive type [27, Chapter 6.3] is a type in Homotopy Type Theory that consists of two points $i_0, i_1 : \mathbb{I}$ and a path $\mathsf{seg} : i_0 =_{\mathbb{I}} i_1$ between them. Its *recursion*, or *non-dependent elimination* principle says: Given

$$Y : \mathcal{U} \tag{9.5}$$

$$y_0 : Y \tag{9.6}$$

$$y_1 : Y \tag{9.7}$$

$$p : y_0 = y_1, \tag{9.8}$$

there exists a function $f : \mathbb{I} \to Y$ such that

$$f(i_0) \equiv y_0 \tag{9.9}$$

$$f(i_1) \equiv y_1 \tag{9.10}$$

$$\mathsf{ap}_f(\mathsf{seg}) = p. \tag{9.11}$$

We abstain from introducing the interval's induction principle (cf. [27, Chapter 6.3] for details). The interval is a contractible type and as such equivalent to the unit type. However, this does not make it entirely boring; it is the *judgmental* equalities that matter. Note that the *computation rules* for the *points* are judgmental (9.9,9.10), while the rule for the path (9.11) is only propositional.

We will now show that $\|\mathbf{2}\|$ can be regarded as the interval.

Theorem 9.1. *For the type $\|\mathbf{2}\|$, the recursion principle of the interval (including the computational behaviour) is derivable using 9.1, and the induction principle follows from 9.2.*

Proof. We only show that the recursion principle is derivable, which will be sufficient for the proceeding developments. The induction principle can be derived very similarly. We need to show that, under the assumptions 9.5-9.8, there is a function $f : \|\mathbf{2}\| \to Y$ such that

$$f(|0_{\mathbf{2}}|) \equiv y_0 \tag{9.12}$$
$$f(|1_{\mathbf{2}}|) \equiv y_1 \tag{9.13}$$
$$\mathsf{ap}_f(\mathsf{h}_{\mathsf{tr}|0_{\mathbf{2}}|,|1_{\mathbf{2}}|}) = p. \tag{9.14}$$

We define

$$g : \mathbf{2} \to \Sigma_{y:Y}\, y_0 = y \tag{9.15}$$
$$g(0_{\mathbf{2}}) :\equiv (y_0, \mathsf{refl}) \tag{9.16}$$
$$g(1_{\mathbf{2}}) :\equiv (y_1, p). \tag{9.17}$$

As $\Sigma_{y:Y}\, y_0 = y$ is contractible, g can be lifted to a function $\overline{g} : \|\mathbf{2}\| \to \Sigma_{y:Y}\, y_0 = y$, and we define $f :\equiv \pi_1 \circ \overline{g}$. It is easy to check that f has indeed the required judgmental properties 9.12 and 9.13. has the required properties. The propositional equality 9.14 is only slightly more difficult: First, using the definition of f and a standard functoriality property of ap [27, Lemma 2.2.2 (iii)], we observe that $\mathsf{ap}_f(\mathsf{h}_{\mathsf{tr}|0_{\mathbf{2}}|,|1_{\mathbf{2}}|})$ may be written as

$$\mathsf{ap}_{\pi_1}(\mathsf{ap}_{\overline{g}}(\mathsf{h}_{\mathsf{tr}|0_{\mathbf{2}}|,|1_{\mathbf{2}}|})). \tag{9.18}$$

But here, the path $\mathsf{ap}_{\overline{g}}(\mathsf{h}_{\mathsf{tr}|0_{\mathbf{2}}|,|1_{\mathbf{2}}|})$ is an equality in the contractible type $(y_0, refl) = (y_1, p)$ (note that both terms inhabit a contractible type themselves) and thereby unique. In particular, it is (propositionally) equal to the path which is built out of two components, the first of which can be chosen to be p (the second component can then be taken to be a canonically constructed inhabitant of $p_*(\mathsf{refl}) = p$). □

9.2. **Function Extensionality.** It is known that the interval $\mathbb{I}$ with its judgmental computation rules implies function extensionality. We may therefore conclude that propositional truncation is sufficient as well.

Lemma 9.2 (Shulman [24])**.** *In a type theory with $\mathbb{I}$ and the judgmental η-law for functions (which we assume), function extensionality is derivable.*

Proof. Assume X, Y are types and $f, g : X \to Y$ are functions with the property $h : \forall x.\, f(x) = g(x)$. Using the recursion principle of $\mathbb{I}$, we may then define a family

$$k : X \to \mathbb{I} \to Y \tag{9.19}$$

of functions, indexed over X, such that $k(x, i_0) \equiv f(x)$ and $k(x, i_0) \equiv g(x)$ for all $x : X$; of course, we use $h(x)$ as the required family of paths. Switching the arguments gives a function

$$k' : \mathbb{I} \to X \to Y \tag{9.20}$$

with the property that $k'(i_0) \equiv f$ and $k'(i_1) \equiv g$ (by η for functions), and thereby $\mathsf{ap}_{k'}(\mathsf{seg}) : f = g$. □

The combination of Theorem 9.1 and Lemma 9.2 implies:

Corollary 9.3. *From propositional truncation with judgmental β and judgmental η for functions, function extensionality can be derived.* □

9.3. **Judgmental Factorization.** The judgmental computation rule of $\|-\|$ also allows us to factor any function *judgmentally* through the propositional truncation as soon as it can be factored in any way. This observation is inspired by and a generalization of the fact that $\|\mathbf{2}\|$ satisfies the judgmental properties of the interval (Theorem 9.1).

Theorem 9.4. *Any (non-dependent) function that factors through the propositional truncation can be factored judgmentally: assume types X, Y and a function $f : X \to Y$ between them. Assume that there is $\overline{f} : \|X\| \to Y$ such that*

$$h : \forall(x : X).\, f(x) = \overline{f}(|x|). \tag{9.21}$$

Then, we can construct a function $f' : \|X\| \to Y$ such that, for all $x : X$, we have

$$f(x) \equiv f'(|x|), \tag{9.22}$$

which means that the type $\forall(x : X).\, f(x) = f'(|x|)$ is inhabited by the function that is constantly refl.

Proof. We define a function

$$g : X \to \Pi_{z:\|X\|}\, \Sigma_{y:Y}\, y = \overline{f}(z) \tag{9.23}$$

$$g(x) :\equiv \lambda z.\, \Big(f(x), h(x) \centerdot \mathsf{ap}_{\overline{f}}(\mathsf{h}_{\mathsf{tr}|x|,z})\Big) \tag{9.24}$$

By function extensionality and the fact that singleton types are contractible, the codomain of g is contractible, and thus, we can lift g and get

$$\overline{g} : \|X\| \to \Pi_{z:\|X\|}\, \Sigma_{y:Y}\, y = \overline{f}(z). \tag{9.25}$$

We define

$$f' :\equiv \lambda z : \|X\|.\pi_1(\overline{g}\, z\, z) \tag{9.26}$$

and it is immediate to check that f' has the required properties. □

Note that in the above argument we have only used 9.1. We have avoided 9.2 by introducing the variable z in (9.23), which is essentially a duplication of the first argument of the function, as it becomes apparent in (9.26).

Furthermore, we have assumed that f is a non-dependent function. The question does not make sense if f is dependent in the sense of $f : \Pi_{x:X} Y(x)$; however, it does for $f : \Pi_{z:\|X\|} Y(z)$. In this case, it seems to be unavoidable to use (9.2), but the above proof still works with minimal adaptions. We state it for the sake of completeness.

Theorem 9.5. *Let X be a type and $Y : \|X\| \to \mathcal{U}$ a type family. Assume we have functions*

$$f : \Pi_{x:X} Y(|x|) \tag{9.27}$$

$$\overline{f} : \Pi_{z:\|X\|} Y(z) \tag{9.28}$$

such that

$$\forall(x : X).\, f(x) =_{Y(|x|)} \overline{f}(|x|). \tag{9.29}$$

Then, we can construct a function $f' : \Pi_{z:\|X\|} B(z)$ with the property that for any $x : X$, we have the judgmental equality

$$f(x) \equiv f'(|x|). \tag{9.30}$$

Proof. Because we allow ourselves to use (9.2) the proof becomes actually simpler than the proof above. This time, we can define

$$g : \Pi_{x:X} \Sigma_{y:Y}\, y = \overline{f}(|x|) \tag{9.31}$$

$$g(x) :\equiv (f(x), h(x))\,. \tag{9.32}$$

Using (9.2), we get

$$\overline{g} : \Pi_{z:\|X\|} \Sigma_{y:Y}\, y = \overline{f}(z). \tag{9.33}$$

Then,

$$\pi_1 \circ \overline{g} \tag{9.34}$$

fulfils the required condition. □

9.4. **An Invertibility Paradox.** An inhabitant of $\|X\|$ shows, in the language of [27], that the type X is *merely inhabited*: it is inhabited, but we do not know more than that. In particular, we do not know an inhabitant of X. It therefore seems to be a reasonable intuition that

$$|-| : X \to \|X\| \tag{9.35}$$

can be understood as an "information hiding" function: a concrete $x : X$ is turned into an anonymous inhabitant $|x| : \|X\|$. While this interpretation is justified to some degree as long as we think of internal properties, it may be misleading from a meta-theoretic point of view.

To make our point clear, we assume Voevodsky's Univalence Axiom (see e.g. [27] for an introduction or [29] for an original reference) which specifies the equality types of the universe. We show that, for a nontrivial class of types, the projection map $|-|$ can be "pseudo-reversed". For example, there is a term that we call $\mathsf{myst}_{\mathbb{N}}$ such that

$$\mathsf{id}' : \mathbb{N} \to \mathbb{N} \tag{9.36}$$

$$\mathsf{id}' :\equiv \mathsf{myst}_{\mathbb{N}} \circ |-| \tag{9.37}$$

type checks and id' is the identity function on $\mathbb{N}$, with a proof

$$\mathsf{p} : \forall(n : \mathbb{N}).\, \mathsf{id}'(n) = n \tag{9.38}$$

$$\mathsf{p} :\equiv \lambda n.\mathsf{refl}_n. \tag{9.39}$$

We think that the possibility to do this is counter-intuitive and surprising. The term $\mathsf{myst}_{\mathbb{N}}$ seems to contradict the intuition that $|-|$ does not make any distinction between elements of $\mathbb{N}$; it sends any such inhabitant to the unique inhabitant of $\|\mathbb{N}\|$. We do indeed have the equalities

$$\mathsf{myst}_{\mathbb{N}}(|0|) \equiv 0 \tag{9.40}$$

$$\mathsf{myst}_{\mathbb{N}}(|1|) \equiv 1, \tag{9.41}$$

and the fact that these are not only propositional, but even judgmental, makes it even stranger. As we know $|0| =_{\|\mathbb{N}\|} |1|$, it might seem that we could prove $0 =_{\mathbb{N}} 1$ from the equations above. Of course, this is not the case. The sketched proof of $0 =_{\mathbb{N}} 1$ would work if the type of $\mathsf{myst}_{\mathbb{N}}$ (which we have not talked about yet) was $\|\mathbb{N}\| \to \mathbb{N}$, but it is not that simple. We show the construction to see what happens. For further discussion, see the homotopy type theory blog entry by the first named author [13], where this result was presented originally.

First, let us state two useful general definitions:

Definition 9.6 (Pointed Types [27, Definition 2.1.7])**.** A *pointed type* is a pair (X, x) of a type $X : \mathcal{U}$ and an inhabitant $x : X$. We write $\mathcal{U}_\bullet$ for the type of pointed types,

$$\mathcal{U}_\bullet :\equiv \Sigma_{X:\mathcal{U}}\, X. \tag{9.42}$$

Definition 9.7 (Transitive Type)**.** Given a type X, we call it *transitive* and write $\mathsf{isTransitive}\, X$ if it satisfies

$$\forall(x\, y : X).\, (X, x) =_{\mathcal{U}_\bullet} (X, y). \tag{9.43}$$

This is, of course, where univalence comes into play. It gives us the principle that a type X is transitive if, and only if, for every pair $(x, y) : X \times X$ there is an automorphism $e_{xy} : X \to X$ such that $e_{xy}(x) = y$.

We have the following examples of transitive types:

Example 9.8. Every type with decidable equality is transitive.

This is because decidable equality on X lets us define an endofunction on X which swaps x and y, and leaves everything else constant. Instances for this example include all contractible and, more generally, propositional types, but also our main candidate, the natural numbers $\mathbb{N}$.

Example 9.9. For any pointed type X with elements $x_1, x_2 : X$, the identity type $x_1 =_X x_2$ is transitive. In particular, the *loop space* $\Omega^n(X)$ [27, Definition 2.1.8] is transitive for any pointed type X.

Here, it is enough to observe that, for $p_1, p_2 : x_1 =_X x_2$, the function $\lambda q.q \cdot {p_1}^{-1} \cdot p_2$ is an equivalence with the required property.

As mentioned by Andrej Bauer in a discussion on this result [13], we also have the following:

Example 9.10. Any group [27, Definition 6.11.1] is a transitive type.

As for equality tyes, the reason is that there is an inverse operation, such that the automorphism $\lambda c.c \cdot a^{-1} \cdot b$ maps a to b.

Example 9.11. If X is any type and $Y : X \to \mathcal{U}$ is a family of transitive types, then $\Pi_{x:X} Y(x)$ is transitive.

In particular, $\times$ and $\to$ preserve transitivity of types.

We are now ready to construct myst: Assume that we are given a type X. We can define a map

$$f : X \to \mathcal{U}_\bullet \tag{9.44}$$

$$f(x) :\equiv (X, x). \tag{9.45}$$

If we know a point $x_0 : X$, we may further define

$$\overline{f} : \|X\| \to \mathcal{U}_\bullet \tag{9.46}$$

$$\overline{f}(z) :\equiv (X, x_0). \tag{9.47}$$

If X is transitive, we have

$$\forall(x : X).\, f(x) = \overline{f}(|x|). \tag{9.48}$$

By Theorem 9.4, there is then a function

$$f' : \|X\| \to \mathcal{U}_\bullet \tag{9.49}$$

such that, for any $x : X$, we have

$$f'(|x|) \equiv f(x) \equiv (X, x). \tag{9.50}$$

Let us define

$$\mathsf{myst}_X : \Pi_{z:\|X\|} \pi_1(f'(z)) \tag{9.51}$$

$$\mathsf{myst}_X :\equiv \pi_2 \circ f'. \tag{9.52}$$

At this point, we can see where the paradox comes from. The type of myst_X is *not* just $\|X\| \to X$; however, for any $x : X$, the type of $f'(|x|)$ is *judgmentally* equal to X, and we have $f'(|x|) \equiv x$. This already proves the following:

Theorem 9.12. *Let X be an inhabited transitive type. Then, there is a term myst_X such that the composition*

$$\mathsf{myst}_X \circ |-| : X \to X \tag{9.53}$$

type checks and is equal to the identity, where the proof

$$p : \forall(x : X).\, \mathsf{myst}_X(|x|) =_X x \tag{9.54}$$

$$p(x) :\equiv \mathsf{refl}_x \tag{9.55}$$

it trivial. □

It is tempting to unfold the type expression $\Pi_{z:\|X\|}\, \pi_1(f'(z))$ in order to better understand it. Unfortunately, this is not very feasible as this plain type expression involves the whole proof term f', which, in turn, includes the complete construction of Theorem 9.4.

Note that Theorem 9.12 does *not* mean that the identity function factorizes through $\|X\|$; because, being careful with this notion, this would require a retraction of $|-| : X \to \|X\|$, which we do *not* have. If we are given $x, y : X$, we do know $\mathsf{h}_{\mathsf{tr}|x|,|y|} : |x| =_{\|X\|} |y|$, but we *cannot* conclude

$$\mathsf{ap}_{\mathsf{myst}_X} : \mathsf{myst}_X(|x|) =_X \mathsf{myst}_X(|y|) \tag{9.56}$$

as this does not type check. Instead, we only have

$$\mathsf{apd}_{\mathsf{myst}_X}\left(\mathsf{h}_{\mathsf{tr}|x|,|y|}\right) \;:\; \left(\mathsf{transport}^{\lambda z.\pi_1(f'(z))}(\mathsf{h}_{\mathsf{tr}|x|,|y|}, \mathsf{myst}_X(|x|))\right) =_{\pi_1(f'(|y|))} \mathsf{myst}_X(|y|), \tag{9.57}$$

where apd is the *dependent* version of ap [27, cf. Lemma 2.3.4]. After evaluating (that is, using *judgmental* equalities to simplify some expressions), (9.57) becomes

$$\mathsf{apd}_{\mathsf{myst}_X}\left(\mathsf{h}_{\mathsf{tr}|x|,|y|}\right) \;:\; \left(\mathsf{transport}^{\pi_1 \circ f'}(h_{|x|,|y|}, x)\right) =_X y. \tag{9.58}$$

But this does not look wrong at all any more as $\pi_1 \circ f'$ is an automorphism on X that sends x to y.

Finally, we want to remark that the construction of myst does not need the full strength of Theorem 9.4. The weaker version in which $\overline{f} : \|X\| \to Y$ is replaced by a fixed $y_0 : Y$ is sufficient: in this case, $\overline{f}$ can be understood to be *strictly* constant. This leads to a simplification as the dependent function types in (9.23) and (9.25) can be replaced by their codomains.

It may be helpful to see the whole definition of myst explicitly in this variant, which is also how it was explained originally by the first named author [13]: We define

$$\mathsf{f} : X \to \Sigma_{A:\mathcal{U}_\bullet}\; A =_{\mathcal{U}_\bullet} (X, x_0) \tag{9.59}$$

$$\mathsf{f}(x) :\equiv ((X, x), \mathsf{transitive}_X(x, x_0)), \tag{9.60}$$

where $\mathsf{transitive}$ is the proof that A is transitive. The function f in (9.44) is then simply the composition $\pi_1 \circ \mathsf{f}$. As the codomain of f is a singleton type, it is contractible (cf. Definition 2.1) and thereby propositional (let us write h for the proof thereof). Hence, we get

$$\mathsf{f}' : \|X\| \to \Sigma_{A:\mathcal{U}_\bullet}\; A =_{\mathcal{U}_\bullet} (X, x_0) \tag{9.61}$$

$$\mathsf{f}' :\equiv \mathsf{rec}_{\mathsf{tr}}\left(\Sigma_{A:\mathcal{U}_\bullet}\; A =_{\mathcal{U}_\bullet} (X, x_0)\right)\; h\; \mathsf{f}. \tag{9.62}$$

We could now define myst'_X to be

$$\mathsf{myst}'_X : \Pi_{\|X\|}\, \pi_1 \circ \pi_1 \circ \mathsf{f}' \tag{9.63}$$

$$\mathsf{myst}'_X :\equiv \pi_2 \circ \pi_1 \circ \mathsf{f}' \tag{9.64}$$

which has the same property as (9.52), even though it is not judgmentally the same term.

10. Conclusion and Open Problems

In this article, generalizations of Hedberg's Theorem have lead us to an exploration of what we call *weakly constant functions*. The attribute *weakly* indicates that higher coherence conditions of such a constancy proof are missing. As a consequence, it is not possible to derive a function $\|X\| \to Y$ from a weakly constant function $X \to Y$, but we have shown how to do this in several non-trivial special cases. Most interesting is certainly the case of endofunctions. A weakly constant endofunction can always be factorized through the propositional truncation of its domain. Further, for a given X, the type which says that every constant endofunction on X has a fixed point is propositional, enabling us to use it as a notion of anonymous inhabitance $\langle\langle X \rangle\rangle$, and we have argued that it lies strictly in between of $\neg\neg X$ and $\|X\|$.

There are two questions for which we have not given an answer. The first is: Is weak propositional truncation definable in Martin-Löf Type Theory? This is commonly believed to not be the case. However, the standard models do have propositional truncation, making it hard to find a concrete proof. Moreover, populatedness, a similar notion of anonymous existence, is definable.

Our second question is the continuation of what we have discussed in the first half of Section 5: Is it possible to derive a (constructive or homotopical) taboo from the assumption that every weakly constant function can be factorized through $\|-\|$? More precisely, does the assumption

$$\forall(X\,Y : \mathcal{U}).\, \forall(f : X \to Y).\, \mathsf{const}\, f \to \|X\| \to Y \tag{10.1}$$

allow us to derive a "suspicious" statement, such as UIP for all types, some form of the axiom of choice (cf. [27, Chapter 3.8]), or some form of excluded middle? For the assumption that every type is collapsible, we have established a corresponding result in Section 6: It implies that all equalities are decidable.

One example of a weakly constant function for which we do not know the status of its factorizability is the function (9.44) which maps, for a given transitive type X, any point $x : X$ to $(X, x) : \mathcal{U}_\bullet$. Being able to factorize it amounts to knowing an inhabitant of

$$\forall(X : \mathcal{U}).\, (\|X\| \times \mathsf{isTransitive}\, X) \to \Sigma_{A:\mathcal{U}}\, A \times (X \to A =_{\mathcal{U}} X). \tag{10.2}$$

We can use the functoriality of $\|-\|$ to see that (10.2) implies

$$\forall(X : \mathcal{U}).\, (\|X\| \times \mathsf{isTransitive}\, X) \to \Sigma_{A:\mathcal{U}}\, A \times \|A =_{\mathcal{U}} X\|. \tag{10.3}$$

The latter statement can be read as: Given a transitive merely inhabited type, can we find an inhabited type that is merely equal to the first? Finally, we can ask one more short question if we drop the transitivity condition and arrive at

$$\forall(X : \mathcal{U}).\, \|X\| \to \Sigma_{A:\mathcal{U}}\, A \times \|A =_{\mathcal{U}} X\| : \tag{10.4}$$

If we have exact knowledge of a type and mere knowledge about an inhabitant, can we "trade" it for mere knowledge of (the structure of) the type and exact knowledge about an inhabitant? We do not expect that (10.2), (10.3) or (10.4) is derivable, but does any of these three assumptions allow us to conclude a taboo?

Acknowledgement

The first named author would like to thank Paolo Capriotti, Ambrus Kaposi, Nuo Li and especially Christian Sattler for fruitful discussions.

References

[1] T. Altenkirch, T. Coquand, M. Escardó, and N. Kraus. Generalizations of Hedberg's theorem (Agda file), 2012/2013. Available at the third-named author's institutional webpage.
[2] Thorsten Altenkirch, Thomas Anberrée, and Nuo Li. Definable Quotients in Type Theory. 2011.
[3] Steve Awodey and Michael A Warren. Homotopy theoretic models of identity types. Technical Report arXiv:0709.0248, Sep 2007.
[4] Steven Awodey and Andrej Bauer. Propositions as [types]. *Journal of Logic and Computation*, 14(4):447–471, 2004.
[5] Paolo Capriotti and Nicolai Kraus. Eliminating higher truncations via constancy. in preparation, 2014.
[6] 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 Nurpl Proof Development System*. Prentice-Hall, NJ, 1986.
[7] Radu Diaconescu. Axiom of choice and complementation. *Proc. Amer. Math. Soc.*, 51:176–178, 1975.
[8] M. P. Fourman and A. Ščedrov. The "world's simplest axiom of choice" fails. *Manuscripta Math.*, 38(3):325–332, 1982.
[9] Michael Hedberg. A coherence theorem for martin-löf's type theory. *Journal of Functional Programming*, 8(4):413–436, 1998.
[10] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *In Venice Festschrift*, pages 83–111. Oxford University Press, 1996.
[11] William A. Howard. The formulae-as-types notion of construction. In J. Roger Seldin, Jonathan P.; Hindley, editor, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. original paper manuscript from 1969.
[12] N. Kraus. A direct proof of Hedberg's theorem, March 2012. Blog post at homotopytypetheory.org.
[13] Nicolai Kraus. The truncation map $| - | : N \to ||N||$ is nearly invertible, October 2013. Blog post at homotopytypetheory.org.
[14] Nicolai Kraus, Martın Escardó, Thierry Coquand, and Thorsten Altenkirch. Generalizations of hedbergs theorem. In Masahito Hasegawa, editor, *11th International Conference, Typed Lambda Calculus and Applications 2013, Eindhoven, The Netherlands, June 26–28, 2013. Proceedings*, volume 7941 of *Lecture Notes in Computer Science*, pages 173–188. Springer Berlin Heidelberg, 2013.
[15] Nicolai Kraus, Martın Escardó, Thierry Coquand, and Thorsten Altenkirch. Notions of anonymous existence in martin-löf type theory (Agda formalization), 2014. Available at the first-named author's institutional webpage.
[16] Per Martin-Löf. An intuitionistic theory of types: predicative part. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73, Proceedings of the Logic Colloquium*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, 1975.
[17] Per Martin-Löf. Constructive mathematics and computer programming. In L. Jonathan Cohen, Jerzy o, Helmut Pfeiffer, and Klaus-Peter Podewski, editors, *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982.
[18] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.
[19] Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.
[20] R. Mines, F. Richman, and W. Ruitenberg. *A Course in constructive algebra*. Universitext. Springer-verlag, New York, 1988.
[21] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology and Göteborg University, Göteborg, Sweden, 2007.

[22] Erik Palmgren. Proof-relevance of families of setoids and identity in type theory. *Arch. Math. Log.*, 51(1-2):35–47, 2012.
[23] Christine Paulin-Mohring. Inductive Definitions in the System Coq - Rules and Properties. In Marc Bezem and Jan Friso Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, 1993.
[24] Michael Shulman. An interval type implies function extensionality (blog post), 2011.
[25] Thomas Streicher. Investigations into intensional type theory, 1993. Habilitationsschrift, Ludwig-Maximilians-Universität München.
[26] The HoTT and UF community. HoTT github repository. Available online.
[27] The Univalent Foundations Program UF. *Homotopy type theory: Univalent foundations of mathematics*. first edition, 2013. Available online at homotopytypetheory.org/book.
[28] UF community. Homotopy type theory (mailing list).
[29] V. Voevodsky. The equivalence axiom and univalent models of type theory. *Talk at CMU*.
[30] Vladimir Voevodsky. Coq library. Availabe at the author's institutional webpage.