



Partial Univalence in n-truncated Type Theory

Downloaded from: <https://research.chalmers.se>, 2026-03-09 07:01 UTC

Citation for the original published paper (version of record):

Sattler, C., Vezzosi, A. (2020). Partial Univalence in n-truncated Type Theory. ACM International Conference Proceeding Series: 807-819. <http://dx.doi.org/10.1145/3373718.3394759>

N.B. When citing this work, cite the original published paper.



Partial Univalence in n -truncated Type Theory

Christian Sattler

Department of Computer Science and Engineering
Chalmers University of Technology
Sweden
sattler@chalmers.se

Andrea Vezzosi

Department of Computer Science
IT University of Copenhagen
Denmark
avez@itu.dk

Abstract

It is well known that univalence is incompatible with uniqueness of identity proofs (UIP), the axiom that all types are h-sets. This is due to finite h-sets having non-trivial automorphisms as soon as they are not h-propositions.

A natural question is then whether univalence restricted to h-propositions is compatible with UIP. We answer this affirmatively by constructing a model where types are elements of a closed universe defined as a higher inductive type in homotopy type theory. This universe has a path constructor for simultaneous "partial" univalent completion, i.e., restricted to h-propositions.

More generally, we show that univalence restricted to $(n - 1)$ -types is consistent with the assumption that all types are n -truncated. Moreover we parametrize our construction by a suitably well-behaved container, to abstract from a concrete choice of type formers for the universe.

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

Keywords: homotopy type theory, cubical type theory, univalence

ACM Reference Format:

Christian Sattler and Andrea Vezzosi. 2020. Partial Univalence in n -truncated Type Theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20)*, July 8–11, 2020, Saarbrücken, Germany. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3373718.3394759>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. *LICS '20*, July 8–11, 2020, Saarbrücken, Germany

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-7104-9/20/07...\$15.00
<https://doi.org/10.1145/3373718.3394759>

1 Introduction, Motivation, and Overview

Martin-Löf type theory [20] (MLTT) is a formal system useful both for dependently typed programming and as a foundation for the development of mathematics. It is the basis of proof assistants like Agda, Coq, Idris, Lean.

Homotopy type theory (HoTT) is a variation born out of the observation that equality proofs in MLTT behave like paths in homotopy theory [6]. A major focus is then to characterize the exact nature of equality for each type, filling some gaps left underspecified by MLTT by taking inspiration from the connection to spaces up to homotopy.

Central is Voevodsky's univalence axiom, stating that equalities of types corresponds to equivalence of types. From univalence other extensionality principles follow, like function and propositional extensionality: equality of functions corresponds to pointwise equality, and equality of propositions corresponds to logical equivalence.

Another important contribution is the introduction of higher inductive types (HITs), which generalize inductive types by not only allowing elements of the type but also equalities between them to be inductively generated. A general example is taking the quotient of a type by a relation, other examples are finite and countable powerset types [13, 27], ordinal notations [21], syntax of type theory up to judgemental equality [2], other forms of colimits, and types of spaces for synthetic homotopy theory [26].

HoTT also brought attention to a classification of types based on the complexity of their equality type. We say that a type is (-2) -truncated or *contractible* if it is equivalent to the unit type, we say a type A is $(n + 1)$ -truncated when for any $x, y : A$, the equality type $x =_A y$ is n -truncated. In particular (-1) -truncated types are referred to as *h-propositions*, and are those for which any two elements are equal, while 0 -truncated types, whose equality types are h-propositions, are called *h-sets*.

The h-sets are the notion of set of homotopy type theory, and where most constructions will belong when using HoTT as a foundation for set-based mathematics or to reason about programs. Restricting oneself to types whose equality type is an h-proposition also avoids having to stipulate coherence conditions between different ways of proving the same equality. Such coherence conditions might be arbitrarily complex and not necessarily expressible within HoTT itself [17].

It would be tempting then, at least for these applications, to assume that every type is an h-set, i.e., the uniqueness of

identity proofs (UIP). In the case of HITs, e.g. for set quotients, an explicit equality constructor can be included to impose the desired truncation level. However we are forced to step outside the h-sets when considering them collectively as a type, which we call the universe of h-sets, $\mathcal{U}^{\leq 0}$. In fact, by univalence, equalities in $\mathcal{U}^{\leq 0}$ correspond to isomorphisms between the equated h-sets, of which in general there are more than one. This is often unfortunate because of, e.g. the need to define sets by induction on a set quotient, or the lack of a convenient type that could take the role of a Grothendieck universe when formalizing categorical semantics in sets or presheaves.

The counterexample of $\mathcal{U}^{\leq 0}$ however does not apply to univalence restricted to h-propositions, i.e. proposition extensionality, since any two proofs of logical equivalence between two propositions can be proven equal. Moreover results about set-truncated HITs often rely on propositional extensionality when defining a map into h-propositions by induction. One example is effectiveness of quotients, i.e., that equalities $[a] =_{A/R} [b]$ between two representative of an equivalence class correspond to proofs of relatedness $R(a, b)$.

In this paper we show, for the first time, that UIP is consistent with univalence for h-propositions, and more generally that, for $n \geq 0$, the assumption that every type is n -truncated is consistent with univalence restricted to $(n - 1)$ -truncated types (Corollary 4.8). We refer to this as *partial univalence*. We note that the result cannot be improved to include univalence for n -truncated types, as that would imply univalence for all types, and then we could prove that the n -th universe is not an n -type by the main result of [18].

We stress that existing truncated models such as the set model or groupoid model [15] do not model partial univalence. Although the set model contains a univalent universe of propositions, it is not the case that the set of small sets is univalent when restricted to propositions. Similarly, the groupoid model contains a univalent universe of sets, but the groupoid of small groupoids is not univalent when restricted to h-sets, i.e. groupoids with propositional sets of morphisms.

The main challenge will thus be how to interpret the universes of such a theory. For a fixed collection of type formers, we show how to overcome this in Section 2, where we construct a partially univalent universe through an indexed higher inductive type. In Section 3 we generalize the construction by a signature of type formers given as an indexed container [1]. We then prove the consistency of a partially univalent n -truncated type theory in Section 4. The proof uses a model of HoTT capable of interpreting indexed higher inductive types to derive a model for our theory.

The viability of MLTT as a programming language relies on the canonicity property: every closed term is equal to one in canonical form. While univalence as an axiom interferes with canonicity, cubical type theory [9] has remedied this

by representing equality proofs as paths from an abstract interval type, and made univalence no longer an axiom. We formulate a partially univalent 0-truncated cubical type theory in Section 5. There we also prove that the theory satisfies homotopy canonicity, the property that every closed term is path equal to one in canonical form. We believe that this result establishes an important first step towards a computational interpretation of the theory. In Section 6 we discuss related works and conclude.

A long version of this article is available [24]. It includes appendices that are used and referenced in our development.

1.1 Formalization

We have formalized the main construction, the subject of Sections 2 and 3, in Cubical Agda. Separately, we have formalized [24, Appendix A] (the material on pushouts of monomorphisms) in Agda with univalence as a postulate. The formalizations are available as supplementary material to this article.

2 A 0-truncated Partially Univalent Universe of 0-types

In this section we set up some preliminary definitions and notations. We then provide a simpler case of our main technical result Theorem 3.13, to exemplify the reasoning necessary.

When reasoning internally in type theory, we write \equiv for judgmental equality and $=$ for internal equality using the identity type. Given a type A , we also write the latter as $A(-, -)$. Given $p : a_0 =_A a_1$ and a family B over A , the dependent equality $b_0 =_{B(p)} b_1$ of b_0 and b_1 over p is shorthand for the identity type $p_*(b_0) =_{B(a_1)} b_1$, where $p_*(b_0)$ is the transport of $b_0 : B(a_0)$ to $B(a_1)$ along p . Given a type A and $n \geq -2$, we write $\text{type}_n(A)$ for the type that A is n -truncated. For $n = -2, -1, 0$, we have the usual special cases $\text{isContr}(A)$, $\text{isProp}(A)$, $\text{isSet}(A)$ of A being contractible, propositional, and a set, respectively. All of these types are propositions.

We recall the notion of a univalent family.

Definition 2.1 (Univalence). A family Y over a type X is *univalent* if the canonical map $X(x_0, x_1) \rightarrow Y(x_0) \simeq Y(x_1)$ is an equivalence for all $x_0, x_1 : X$.

We relativize this notion with respect to a property P on types. This is supposed to be an extensional property, in the sense that it should depend only on $Y(x)$, not on the “code” $x : X$. To make this precise, we let Y be a valued in a universe \mathcal{U} and express P as a propositional family over \mathcal{U} .

We use the word universe in a rather weak sense: until we add closure under some type formers, it can refer to an arbitrary type family. Notationwise, universes are distinguished in that we leave the decoding function from elements of \mathcal{U} to types implicit.

Definition 2.2 (Partial univalence). Let X be a type and $Y : X \rightarrow \mathcal{U}$ for some universe \mathcal{U} . Let P be a propositional family over \mathcal{U} . We say that (X, Y) is P -univalent if the restriction of the family Y to the subtype $\sum_{x:X} P(Y(x))$ is univalent.

We also say that X is *partially univalent* or *univalent for P* , leaving Y implicit. For $P \equiv \text{type}_n$, we say that X is *univalent for n -types*. In particular, for $P \equiv \text{isProp}$, we say that X is *univalent for propositions*.

Of particular importance is the case where Y is the identity function. In that case, we say that *the universe \mathcal{U} is P -univalent*.

Lemma 2.3. *Let the universe \mathcal{U} be P -univalent. Then $Y : X \rightarrow \mathcal{U}$ is P -univalent exactly if its restriction to $x : X$ with $P(Y(x))$ is an embedding.*

Proof. Let $x_1, x_2 : X$ with $P(Y(x_1))$ and $P(Y(x_2))$. Consider the commuting diagram

$$\begin{array}{ccc} X(x_1, x_2) & \longrightarrow & \mathcal{U}(Y(x_1), Y(x_2)) \\ & \searrow & \swarrow \\ & Y(x_1) \simeq Y(x_2) & \end{array}$$

Since \mathcal{U} is P -univalent, the right map is an equivalence. By 2-out-of-3, the left map is invertible exactly if the top map is invertible. Quantifying over x_1, x_2 , we obtain the claim. \square

2.1 Partially Univalent Type V of (Small) Sets

Let now $\mathcal{U}^{\leq 0}$ be a univalent universe of sets, meaning its elements decode to 0-truncated types. We wish to define a ‘‘closed’’ 0-truncated universe V with a decoding function $\text{El}_V : V \rightarrow \mathcal{U}^{\leq 0}$ that is univalent for propositions in the sense of Definition 2.2. We illustrate the essential features of our construction by requiring that:

- V contains codes for a fixed family $N : M \rightarrow \mathcal{U}^{\leq 0}$ of elements of $\mathcal{U}^{\leq 0}$ where M is a set.
- V is closed under Π -types (assuming that $\mathcal{U}^{\leq 0}$ is),

The former family can for example include codes for the empty type or the type of Booleans.

Closed universes are typically defined by induction-recursion, simultaneously defining the type V and the function $\text{El}_V : V \rightarrow \mathcal{U}^{\leq 0}$. To model the above closure conditions, one takes:

- given $m : M$, a constructor $\bar{N}(m) : V$ and a clause $\text{El}_V(\bar{N}(m)) \equiv N(m)$.
- given $\bar{A} : V$ and $\bar{B} : \text{El}_V(\bar{A}) \rightarrow V$, a constructor $\bar{\Pi}(\bar{A}, \bar{B}) : V$ and a clause

$$\text{El}_V(\bar{\Pi}(\bar{A}, \bar{B})) \equiv \prod_{(a:\text{El}_V(\bar{A}))} \text{El}_V(\bar{B}(a)),$$

In order to make V univalent for propositions, one could imagine turning this into a *higher inductive-recursive* definition. Given $\bar{A}_i : V$ with $\text{El}_V(\bar{A}_i)$ a proposition for $i \in \{0, 1\}$ and an equality $p : \mathcal{U}^{\leq 0}(\text{El}_V(\bar{A}_0), \text{El}_V(\bar{A}_1))$, one would add a

path constructor $\text{ua}(e) : V(\bar{A}_0, \bar{A}_1)$ with a clause giving an identification of the action of El_V on the path $\text{ua}(e)$ with p .¹ This is the right idea, but there are problems.

- While syntax and semantics of higher inductive types have been analyzed to a certain extent [16, 19] this analysis does not yet extend to the case of induction-recursion. As such, the rules for higher inductive-recursive types have not yet been established and none of the known models of homotopy type theory have been shown to admit them.
- Induction-recursion is known to increase the proof-theoretic strength of the type theory over just (indexed) induction. Thus, we do not wish to assume it in our ambient type theory.

Given a type I , recall that types X with a map $X \rightarrow I$ are equivalent to families over I : in the forward direction, one takes fibers; in the backward direction, one takes the total type. Exploiting this correspondence, the above inductive-recursive definition of V (without path constructor for partial univalence) can be turned into an *indexed inductive* definition of a family inV over $\mathcal{U}^{\leq 0}$. The translation of the constructors for Π -types and M is given in (i) and (ii) of Definition 2.4. The path constructor for partial univalence corresponds to the following: given propositions $A_i : \mathcal{U}^{\leq 0}$ with $w_{A_i} : \text{inV}(A_i)$ for $i \in \{0, 1\}$ and an equality $p : \mathcal{U}^{\leq 0}(A_0, A_1)$, we have a path $\text{ua}(e)$ in the family inV between w_{A_0} and w_{A_1} over p . We contract the path p with one of its endpoints and arrive at the definition below.

Definition 2.4. The family inV over $\mathcal{U}^{\leq 0}$ is defined as the following higher indexed inductive type:

- given $m : M$, a constructor $w_N(m) : \text{inV}(N(m))$,
- given $w_A : \text{inV}(A)$ and $w_B(a) : \text{inV}(B(a))$ for $a : A$ (with implicit $A : \mathcal{U}^{\leq 0}$ and $B : A \rightarrow \mathcal{U}^{\leq 0}$), a constructor $w_{\Pi}(w_A, w_B) : \text{inV}(\prod_{(a:A)} B(a))$,
- given a proposition $X : \mathcal{U}^{\leq 0}$ with $w_0, w_1 : \text{inV}(X)$, a path constructor $\text{ua}(w_0, w_1) : w_0 = w_1$.

We recover V as the total type $V = \sum_{(X:\mathcal{U}^{\leq 0})} \text{inV}(X)$, with El_V given by the first projection.

We may regard inV as a *conditionally* or *partially propositionally truncated* indexed inductive type (see [24, Appendix B]). In this form, it becomes clear that the constructor ua indeed suffices for partial univalence and does not introduce coherence problems: it exactly enforces that the restriction of the family inV to elements decoding to propositions is valued in propositions.

Lemma 2.5. *The type V with $\text{El}_V : V \rightarrow \mathcal{U}^{\leq 0}$ is univalent for propositions.*

Proof. Using Lemma 2.3, we have to show that the restriction of $\text{El}_V : V \rightarrow \mathcal{U}^{\leq 0}$ to $\bar{X} : V$ with $\text{El}_V(\bar{X})$ a proposition is

¹ In this particular case, the clause of El_V for the path constructor amounts to nothing as it is an identification in a propositional type.

an embedding. Unfolding to inV , this says that $\text{inV}(X)$ is propositional for $X : \mathcal{U}^{\leq 0}$ a proposition. This is exactly enforced by the path constructor (iii) in Definition 2.4. \square

It remains to show that V is 0-truncated. For this, we adapt the encode-decode method to characterize the dependent equalities in inV over an equality in $\mathcal{U}^{\leq 0}$.

2.2 Dependent Equalities in inV

In the following, we make use of (homotopy) pushouts. Recall [26, Section 6.8] that the *pushout* of a span $f : A \rightarrow B$ and $g : A \rightarrow C$ of types is the (non-recursive) higher inductive type $B +_A C$ with points constructors $\text{inl}(b) : B +_A C$ for $b : B$ and $\text{inr}(c) : B +_A C$ for $c : C$ and path constructor $\text{glue}(a) : (B +_A C)(\text{inl}(f(b)), \text{inr}(g(c)))$ for $a : A$.

As in [22, Lecture 13], we do not require judgmental β -laws even for point constructors. Thus, pushout types are simply a particular choice of pushout squares, (homotopy) initial cocones under the span $B \leftarrow A \rightarrow C$. We refer to [24, Appendix A] for some key properties of pushouts used in our development.

An important special case is the *join* $X \star Y$ of types X and Y , the pushout of X and Y under $X \times Y$. For a proposition P , the operation $P \star -$ is also known as the *closed modality* associated with P [23, Example 1.8]. We will only use the join in this form. Recall that $X \star Y$ is contractible if X or Y is contractible. In particular, $P \star X$ is contractible if P holds.

Problem 2.6. *Given an equality $p : \mathcal{U}^{\leq 0}(X_0, X_1)$ and $w_i : \text{inV}(X_i)$ for $i \in \{0, 1\}$, we wish to define:*

- a type $\text{Eq}_{\text{inV}}(p, w_0, w_1)$ of codes of equalities over p between w_0 and w_1
- such that $\text{Eq}_{\text{inV}}(p, w_0, w_1)$ is contractible if X_0 (or equivalently X_1) is a proposition.

Construction. By univalence for contractible types, the type of contractible types is contractible. Thus, the goal is contractible if X_0 or X_1 is a proposition.

We perform double induction, first on $w_0 : \text{inV}(X_0)$ and then on $w_1 : \text{inV}(X_1)$. In all path constructor cases, we know that X_0 or X_1 is a proposition. By the above, the goal becomes an equality in a contractible type, so there is nothing to show.

In all point constructor cases, we define

$$\text{Eq}_{\text{inV}}(u, w_0, w_1) \equiv_{\text{def}} \text{isProp}(X_0) \star E$$

where E is an abbreviation for an expression that varies depending on the case. Note that this makes $\text{Eq}_{\text{inV}}(u, w_0, w_1)$ contractible when X_0 is a proposition. The expression E codes structural equality of the top-level constructors.

- For $w_0 \equiv w_N(m_0)$ and $w_1 \equiv w_N(m_1)$, we let E consist of pairs (p_m, c) where $p_m : M(m_0, m_1)$ and c is a proof that $p : \mathcal{U}^{\leq 0}(N(m_0), N(m_1))$ is equal to the action of M on p_m .

- For $w_i \equiv w_{\Pi}(w_{A_i}, w_{B_i})$ with $w_{A_i} : \text{inV}(A_i)$ and $w_{B_i}(a_i) : \text{inV}(B_i(a_i))$ for $a_i : A_i$, all for $i \in \{0, 1\}$, we let E consist of tuples (p_A, e_A, p_B, e_B, c) where:
 - $p_A : \mathcal{U}^{\leq 0}(A_0, A_1)$ with $e_A : \text{Eq}_{\text{inV}}(p_A, w_{A_0}, w_{A_1})$,
 - for $a_0 : A_0$, $a_1 : A_1$, and an equality p_a over p_A between a_0 and a_1 , we have $p_B : \mathcal{U}^{\leq 0}(B_0(a_0), B_1(a_1))$ with $e_B : \text{Eq}_{\text{inV}}(p_B, w_{B_0}(a_0), w_{B_1}(a_1))$.
 - c witnesses that

$$p : \mathcal{U}^{\leq 0}(\prod_{(a_0:A_0)} B_0(a_0), \prod_{(a_1:A_1)} B_1(a_1))$$

is equal to the action of the type forming operation Π on p_A and p_B .

- In the remaining “mixed” cases, we let E be empty. \square

Proposition 2.7. *Given $p : \mathcal{U}^{\leq 0}(X_0, X_1)$ and $w_i : \text{inV}(X_i)$ for $i \in \{0, 1\}$, there is an equivalence between dependent equalities in inV over p between w_0 and w_1 and $\text{Eq}_{\text{inV}}(p, w_0, w_1)$.*

Proof. For the purpose of this proof, it will be convenient to work with a different, but equivalent definition of the expression E in the construction of Problem 2.6 in the case $w_i \equiv w_{\Pi}(w_{A_i}, w_{B_i})$ with $w_{A_i} : \text{inV}(A_i)$ and $w_{B_i}(a_i) : \text{inV}(B_i(a_i))$ for $a_i : A_i$, all for $i \in \{0, 1\}$. Namely, we let E consist of pairs (q, r) as follows.

- The component q is an equality $(A_0, B_0) = (A_1, B_1)$ in the dependent sum $\sum_{(A:\mathcal{U}^{\leq 0})} A \rightarrow \mathcal{U}^{\leq 0}$.
- Inducting on the equality q , we may suppose $A \equiv_{\text{def}} A_0 \equiv A_1$ and $B \equiv_{\text{def}} B_0 \equiv B_1$. The component r is then a triple (e_A, e_B, c) where
 - $e_A : \text{Eq}_{\text{inV}}(\text{refl}_A, w_{A_0}, w_{A_1})$,
 - $e_B(a) : \text{Eq}_{\text{inV}}(\text{refl}_{B(a)}, w_{B_0}(a), w_{B_1}(a))$ for $a : A$,
 - $c : p = \text{refl}$.

The equivalence between this choice of E and the previous one is a straightforward consequence of structural equivalences, splitting up the equality q into components for A and B and distributing them over the components of r .

We follow the encode-decode method as described in [24, Subappendix B.3]. To define

$$w_0 \equiv_{\text{inV}(p)} w_1 \xrightarrow{\text{encode}_{p, w_0, w_1}} \text{Eq}_{\text{inV}}(p, w_0, w_1),$$

we use equality induction on p and the argument, reducing the goal to $\text{encode}'(w) : \text{Eq}_{\text{inV}}(\text{refl}_X, w, w)$ for $w : \text{inV}(X)$. We induct on $x : \text{inV}(X)$.

- For $w \equiv w_N(m) : \text{inV}(N(m))$, we take

$$\text{encode}'(x) \equiv \text{inr}(\text{refl}_m, \text{refl}).$$

- For $w \equiv w_{\Pi}(w_A, w_B) : \text{inV}(\prod_{(a:A)} B(a))$, we take

$$\text{encode}'(x) \equiv \text{inr}$$

$$(\text{refl}_{(A,B)}, (\text{encode}'(w_A), \lambda a. \text{encode}'(w_B(a)), \text{refl})).$$

- In the path constructor case, we have that X is a proposition. Then the goal is a dependent equality in a contractible type.

We now show $\text{encode}_{p, w_0, w_1}^{-1}(e)$ for $e : \text{Eq}_{\text{inV}}(p, w_0, w_1)$. We use double induction on w_0 and w_1 . In all path constructor cases, we know that X_0 or X_1 is a proposition (hence both are). Thus, both source and target of $\text{encode}_{p, w_0, w_1}$ are contractible, so the goal becomes contractible. In all point constructor cases, we have $e : \text{isProp}(X_0) \star E$ where E depends on the particular case. We induct on e . In the case for inl or glue , we have $\text{isProp}(X_0)$, and the goal becomes contractible. For $e \equiv \text{inr}(z)$, we proceed with z according to the point constructor case for w_0 and w_1 .

- For $w_0 \equiv w_N(m_0)$ and $w_1 \equiv w_N(m_1)$, we have $z \equiv (p_m, c)$. By equality induction on $p_m : M(m_0, m_1)$, we may suppose $m \equiv_{\text{def}} m_0 \equiv m_1$ and $p_m \equiv \text{refl}_m$. By equality induction on c , we may then suppose $p \equiv \text{refl}$ and $c \equiv \text{refl}$. We have

$$\begin{aligned} e &\equiv \text{inr}(\text{refl}_m, \text{refl}) \\ &\equiv \text{encode}'(w_N(m)) \\ &\equiv \text{encode}_{\text{refl}_{N(m)}, w_N(m), w_N(m)}(\text{refl}), \end{aligned}$$

showing $\text{encode}_{p, w_0, w_1}^{-1}(e)$.

- For $w_i \equiv w_{\Pi}(w_{A_i}, w_{B_i})$ with $w_{A_i} : \text{inV}(A_i)$ and $w_{B_i}(a_i) : \text{inV}(B_i(a_i))$ for $a_i : A_i$, all for $i \in \{0, 1\}$, we have $z = (q, r)$ as described at the beginning of this proof. By equality induction on $q : (A_0, B_0) = (A_1, B_1)$, we may suppose $A \equiv_{\text{def}} A_0 \equiv A_1$, $B \equiv_{\text{def}} B_0 \equiv B_1$, and $q \equiv_{\text{def}} \text{refl}$. Then $r = (e_A, e_B, c)$. By equality induction c , we may suppose that $p \equiv \text{refl}$ and $c \equiv \text{refl}$. By induction hypothesis, we have

$$\begin{aligned} &\text{encode}_{\text{refl}_A, w_{A_0}, w_{A_1}}^{-1}(e_A), \\ &\text{encode}_{\text{refl}_{B(a)}, w_{B_0}(a), w_{B_1}(a)}^{-1}(e_B(a)) \text{ for } a : A. \end{aligned}$$

By equality induction and function extensionality, we may thus suppose that

$$\begin{aligned} e_A &\equiv \text{encode}_{\text{refl}_A, w_{A_0}, w_{A_1}}(q_A), \\ e_B &\equiv \lambda a. \text{encode}_{\text{refl}_{B(a)}, w_{B_0}(a), w_{B_1}(a)}(q_B(a)) \end{aligned}$$

for some $q_A : w_{A_0} = w_{A_1}$ and $q_B(a) : w_{B_0}(a) = w_{B_1}(a)$ for $a : A$. By equality induction on q_A and q_B (after using function extensionality), we may suppose that $w_A \equiv_{\text{def}} w_{A_0} \equiv w_{A_1}$ and $q_A \equiv \text{refl}$ as well as $w_B \equiv_{\text{def}} w_{B_0} \equiv w_{B_1}$ and $q_B \equiv \lambda a. \text{refl}$. Now we have

$$\begin{aligned} e &\equiv \text{inr}(\text{refl}_{(A, B)}, (\text{encode}'(w_A), \lambda a. \text{encode}'(w_B(a)), \text{refl})) \\ &\equiv \text{encode}'(w_{\Pi}(w_A, w_B)) \\ &\equiv \text{encode}_{\text{refl}_{\Pi(a:A)} B(a), w_{\Pi}(w_A, w_B), w_{\Pi}(w_A, w_B)}(\text{refl}), \end{aligned}$$

showing $\text{encode}_{p, w_0, w_1}^{-1}(e)$.

- In all “mixed” cases, we have $z : 0$. \square

For readers concerned with the length of the above argument, we note the following. In Section 3, we will motivate abstraction that will allow us to reorganize the above argument into smaller, more general pieces.

2.3 V is a set

From our characterization of dependent equality in inV , we obtain a corresponding characterization of equality in V . Given $\bar{X}_i \equiv (X_i, w_i) : V$ for $i \in \{0, 1\}$, we define

$$\text{Eq}_V(\bar{X}_0, \bar{X}_1) \equiv_{\text{def}} \sum_{p : \mathcal{U}^{\leq 0}(X_0, X_1)} \text{Eq}_{\text{inV}}(p, w_0, w_1).$$

Corollary 2.8. *For $\bar{X}_0, \bar{X}_1 : V$, we have*

$$V(\bar{X}_0, \bar{X}_1) \simeq \text{Eq}_V(\bar{X}_0, \bar{X}_1).$$

Proof. Equality types in the type $V \equiv \sum_{(X : \mathcal{U}^{\leq 0})} \text{inV}(X)$ are dependent sums of an equality in $\mathcal{U}^{\leq 0}$ and a dependent equality over it. Thus, the claim is a consequence of Proposition 2.7. \square

Proposition 2.9. *The type V is 0-truncated.*

Proof. Given $\bar{X}_i \equiv (X_i, w_i) : V$ for $i \in \{0, 1\}$, we wish to show $V(\bar{X}_0, \bar{X}_1)$ propositional. By Corollary 2.8, this amounts to showing $\text{Eq}_V(\bar{X}_0, \bar{X}_1)$ propositional. This we show by double induction, first on $w_0 : \text{inV}(X_0)$ and then on $w_1 : \text{inV}(X_1)$. Since the goal is propositional, there is nothing to show in the path constructor cases.

In all point constructor cases, we have

$$\text{Eq}_V(\bar{X}_0, \bar{X}_1) = \sum_{p : \mathcal{U}^{\leq 0}(X_0, X_1)} \text{isProp}(X_0) \star E(p) \quad (1)$$

where $E(p)$ is as in the construction for Problem 2.6, abbreviating an expression depending on the point constructor case (for clarity, we have made the dependency on p explicit). By definition, this join forms a pushout square

$$\begin{array}{ccc} \text{isProp}(X_0) \times E(p) & \longrightarrow & \text{isProp}(X_0) \\ \downarrow & & \downarrow \\ E(p) & \longrightarrow & \text{isProp}(X_0) \star E(p). \end{array} \quad (2)$$

The dependent sum over a fixed type preserves pushout squares in its remaining argument (abstractly, because it is a higher functor left adjoint to weakening). From (1) and (2), we thus obtain the following pushout square:

$$\begin{array}{ccc} \sum_{(p : \mathcal{U}^{\leq 0}(X_0, X_1))} \text{isProp}(X_0) \times E(p) & \longrightarrow & \sum_{(p : \mathcal{U}^{\leq 0}(X_0, X_1))} \text{isProp}(X_0) \\ \downarrow & & \downarrow \\ \sum_{(p : \mathcal{U}^{\leq 0}(X_0, X_1))} E(p) & \longrightarrow & \text{Eq}_V(\bar{X}_0, \bar{X}_1). \end{array} \quad (3)$$

Since $\mathcal{U}^{\leq 0}$ is univalent, the type $\mathcal{U}^{\leq 0}(X_0, X_1)$ is propositional if $\text{isProp}(X_0)$. From this, we see that the span in (3) is a (homotopy) product span. By invariance of pushouts under equivalence, it follows that $\text{Eq}_V(\bar{X}_0, \bar{X}_1)$ is equivalent to the join

$$(\mathcal{U}^{\leq 0}(X_0, X_1) \times \text{isProp}(X_0)) \star \left(\sum_{p : \mathcal{U}^{\leq 0}(X_0, X_1)} E(p) \right). \quad (4)$$

We now apply [24, Lemma A.9]: to show that this join is propositional, it suffices to show that each of its factors is propositional.² Since $\mathcal{U}^{\leq 0}(X_0, X_1)$ is propositional if we have $\text{isProp}(X_0)$, and the latter is propositional, then their product is propositional as well.

It remains to show that $T \equiv_{\text{def}} \sum_{(p:\mathcal{U}^{\leq 0}(X_0, X_1))} E(p)$ is a proposition. For this, we argue according to the current point constructor case, recalling the corresponding definition of $E(p)$ from Problem 2.6.

- In the case $w_0 \equiv w_N(m_0)$ and $w_1 \equiv w_N(m_1)$, we have

$$\begin{aligned} T &\equiv \sum_{(p:\mathcal{U}^{\leq 0}(X_0, X_1))} \sum_{(p_m:\mathcal{M}(m_0, m_1))} (p = \text{ap}_N(p_m)) \\ &\simeq M(m_0, m_1), \end{aligned}$$

a proposition since M was assumed a set.

- In the case $w_i \equiv w_{\Pi}(w_{A_i}, w_{B_i})$ with $w_{A_i} : \text{inV}(A_i)$ and $w_{B_i}(a_i) : \text{inV}(B_i(a_i))$ for $a_i : A_i$ for $i \in \{0, 1\}$, recall that T consists of tuples $(p, p_A, e_A, p_B, e_B, c)$ with types as in the construction of Problem 2.6. We contract the equality c with its endpoint p . What remains is equivalent to the dependent sum of:
 - $(p_A, e_A) : \text{Eq}_V(\bar{A}_0, \bar{A}_1)$ where $\bar{A}_i \equiv_{\text{def}} (A_i, w_i)$,
 - for $a_0 : A_0, a_1 : A_1$, and a dependent equality p_A over p_A between a_0 and a_1 :

$$(p_B, e_B) : \text{Eq}_V(\bar{B}_0(a_0), \bar{B}_1(a_1))$$

where $\bar{B}_i(a_i) \equiv_{\text{def}} (B_i(a_i), w_i(a_i))$.

Both $\text{Eq}_V(\bar{A}_0, \bar{A}_1)$ and $\text{Eq}_V(\bar{B}_0(a_0), \bar{B}_1(a_1))$ (the latter for all a_0, a_1) are $(n-1)$ -truncated by induction hypothesis. The claim now follows by closure of $(n-1)$ -truncated types under dependent sums and dependent products (with arbitrary domain).

- In the remaining, “mixed” cases, we have

$$T \equiv \sum_{p:\mathcal{U}^{\leq 0}(X_0, X_1)} \perp \simeq \perp,$$

which is propositional. \square

3 n -truncated Partially Univalent Universes of n -types

To obtain the main result of this paper, we need to generalize the constructions of the previous section to a partially univalent n -truncated universe of n -types rather than sets, and to a universe closed under more type formers. For the sake of generality, we will also build a universe that is P -univalent for an arbitrary proposition P , although the n -truncatedness result will need $P(X)$ to imply $\text{type}_{(n-1)}(X)$.

This section only makes use of univalence for $(n-1)$ -types.

²Alternatively, we could appeal to the fact that closed modalities are left exact, hence preserve truncation levels [23].

3.1 Indexed Containers and Preservation of Truncation

To abstract from a particular choice of type formers, we will parametrize our universe by a signature of them represented by an indexed container [1], as is done for indexed W -types.

We recall here the precise definitions of indexed container and its extension that we will use in the rest of the paper.

Definition 3.1 (Indexed container). Given a type I , an I -indexed container is a pair (S, Pos) of a type family S over I and a type family Pos over $\sum_{(i:I)} S(i) \times I$.

Definition 3.2 (Extension of a container). Let (S, Pos) be an I -indexed container. Its extension $\text{Ext}_{S, \text{Pos}}$ takes a family F over I and produces another:

$$\text{Ext}_{S, \text{Pos}}(F, i) = \sum_{(s:S(i))} \prod_{(j:I)} \text{Pos}(i, s, j) \rightarrow F(j)$$

Given $(s, t) : \text{Ext}_{S, \text{Pos}}(F, i)$ we will write $t_Y(p)$ for $t(Y)(p)$.

In the universe construction we will use a $\mathcal{U}^{\leq n}$ -indexed container, here we demonstrate by example that they not only cover the type formers considered in Section 2, but also ones with a more complex signature like (truncated) pushouts.

Example 3.3 (Nullary type formers). Given a fixed family of types $N : M \rightarrow \mathcal{U}^{\leq n}$ we define a container with empty positions:

$$\begin{aligned} S(X) &= \sum_{(m:M)} (X = N(m)) \\ \text{Pos}(_, _, _) &= \perp \end{aligned}$$

Example 3.4 (Π -types). The signature for Π -types can be represented by a $\mathcal{U}^{\leq n}$ -indexed container where both S and Pos are given by indexed inductive types with constructors:

- given $A : \mathcal{U}^{\leq n}$ and $B : A \rightarrow \mathcal{U}^{\leq n}$ a constructor $\pi(A, B) : S(\prod_{(x:A)} B(x))$.

and with $s \equiv \pi(A, B)$ and $X \equiv \prod_{(a:A)} B(a)$:

- a constructor $\text{pos}_A : \text{Pos}(X, s, A)$
- given $a : A$ a constructor $\text{pos}_B : \text{Pos}(X, s, B(a))$

Example 3.5 (Truncated pushouts). Pushouts truncated to be n -types can also be represented as a $\mathcal{U}^{\leq n}$ -indexed container:

- given $A_i : \mathcal{U}^{\leq n}$ for $i \in \{0, 1, 2\}$ and $f : A_0 \rightarrow A_1$ and $g : A_0 \rightarrow A_2$ a constructor $\text{po}(f, g) : S(A_1 +_{A_0}^n A_2)$,
- for each $i \in \{0, 1, 2\}$ a constructor $\text{pos}_i : \text{Pos}(A_1 +_{A_0}^n A_2, \text{po}(f, g), A_i)$.

More generally, this works for arbitrary HITs with an additional constructor ensuring n -truncatedness.

To establish the n -truncatedness of the universe we will need to know that the extension of the container $\text{Ext}_{S, P}(F, i)$ preserves the truncation level of the family F . We cannot however just ask for $\text{type}_n(\sum_{(i:I)} \text{Ext}_{S, P}(F, i))$ to hold whenever $\text{type}_n(\sum_{((i:I))} F(i))$ holds, as the latter would already be the whole result when $F \equiv \text{inV}$. We extract then the following condition from what is needed during the induction in the proof of Theorem 3.13.

Definition 3.6 (Retaining n -truncatedness). An I -indexed container (S, Pos) retains n -truncation, if for any family F over I , any $i_b : I$ and element of the extension $(s_b, t_b) : \text{Ext}_{S, Pos}(F, i_b)$ for $b \in \{0, 1\}$ we have that

$$\prod_{(j_0, j_1 : I)} \prod_{(p_0 : \text{Pos}(i_0, s_0, t_0), p_1 : \text{Pos}(i_1, s_1, t_1))} \text{type}_{(n-1)}(\sum_{(q : j_0 = j_1)} t_{j_0}(p_0) \equiv_{F(q)} t_{j_1}(p_1))$$

implies

$$\text{type}_{(n-1)}((i_0, s_0, t_0) \equiv_{\sum_{(i : I)} \text{Ext}_{S, Pos}(F, i)} (i_1, s_1, t_1))$$

Example 3.7 (Signatures retaining n -truncatedness). Examples 3.3 to 3.5 all retain n -truncatedness. The case of nullary type formers is trivial. As mentioned the case for Π -types follows the reasoning in Proposition 2.9. For truncated pushouts we can observe that $(X, (s, t))$ of type $\sum_{(X : \mathcal{U}^{\leq n})} \text{Ext}_{S, Pos}(F, X)$ is equivalent to the following data:

- $X : \mathcal{U}^{\leq n}$
- for $i \in \{0, 1, 2\}$, both $A_i : \mathcal{U}^{\leq n}$ and $w_i : F(A_i)$
- $f : A_0 \rightarrow A_1$ and $g : A_0 \rightarrow A_2$
- $q : X = A_1 +_{A_0}^n A_2$

then X and q form a contractible pair, the types of f and g are n -truncated by construction, so we only have to worry about the (A_i, w_i) pairs. But since the w_i are obtained from t , those components are handled by the premise given to us.

Coproducts of such containers also retain n -truncatedness as their extension will correspond to the sum of the extensions, which means we can collect multiple type formers into a single n -truncatedness preserving indexed container.

3.2 A P -univalent n -truncated Universe of n -truncated Types

Now we have everything in place to provide the final version of our universe

$$\mathbb{V} = \sum_{X : \mathcal{U}^{\leq n}} \text{inV}_{S, Pos}^{n, P}(X)$$

with $\text{El}_{\mathbb{V}} : \mathbb{V} \rightarrow \mathcal{U}^{\leq n}$ given by first projection. We will often omit the sub- and sup- scripts on inV as they will be clear from context.

The family inV is defined as follows. In analogy with the indexed W -type $W_{S, Pos}$, which one would use for an ordinary closed universe, we use the P -propositional indexed W -type $\text{inV} \equiv_{\text{def}} W_{S, Pos}^P$ ([24, Definition B.3]). The theory of partially propositional indexed W -types is developed in [24, Appendix B]. For convenience, we give here the explicit definition as an indexed higher inductive type.

Definition 3.8. Given a $\mathcal{U}^{\leq n}$ -indexed container (S, Pos) , the family inV over $\mathcal{U}^{\leq n}$ is defined as the higher inductive type generated by the following constructors:

- (i) given $c : \text{Ext}_{S, Pos}(\text{inV}, X)$, a constructor $\text{tcon}(c) : \text{inV}(X)$,
- (ii) given $X : \mathcal{U}^{\leq n}$ with $P(X)$, and $w_0, w_1 : \text{inV}(X)$, a path constructor $\text{ua}(w_0, w_1) : w_0 \equiv_{\text{inV}(X)} w_1$.

Lemma 2.5 generalizes to the new setting.

Lemma 3.9. Let (S, Pos) be a $\mathcal{U}^{\leq n}$ -indexed container, and P a family of propositions over $\mathcal{U}^{\leq n}$. The type \mathbb{V} with $\text{El}_{\mathbb{V}} : \mathbb{V} \rightarrow \mathcal{U}^{\leq n}$ is P -univalent. \square

We unfold here the definition of codes for equality in $W_{S, Pos}^P$ of [24, Subappendix B.4].

Problem 3.10. Given an equality $p : \mathcal{U}^{\leq n}(X_0, X_1)$ and $w_i : \text{inV}(X_i)$ for $i \in \{0, 1\}$, we define:

- a type $\text{Eq}_{\text{inV}}(p, w_0, w_1)$ of codes of equalities between w_0 and w_1 over p , as in Figure 1.
- such that $\text{Eq}_{\text{inV}}(p, w_0, w_1)$ is contractible if $P(X_0)$.

Construction. The definition proceeds by double induction on w_0 and w_1 , defining the pair of $\text{Eq}_{\text{inV}}(p, w_0, w_1)$ and its conditional contractibility in one go. Given $P(X_0)$ the case when both w_i are built with tcon is contractible because it's a join with an inhabited proposition. When either w_0 or w_1 is built by ua we again have by univalence that the type of contractible types is contractible. \square

From [24, Subappendix B.4], we have the following result.

Proposition 3.11. Given $p : \mathcal{U}^{\leq n}(X_0, X_1)$ and $w_i : \text{inV}(X_i)$ for $i \in \{0, 1\}$, there is an equivalence

$$w_0 \equiv_{\text{inV}(p)} w_1 \simeq \text{Eq}_{\text{inV}}(p, w_0, w_1). \quad \square$$

As in Section 2 we define

$$\text{Eq}_{\mathbb{V}}(p, \bar{X}_0, \bar{X}_1) \equiv_{\text{def}} \sum_{p : \mathcal{U}^{\leq n}(X_0, X_1)} \text{Eq}_{\text{inV}}(p, w_0, w_1).$$

for $\bar{X}_i \equiv (X_i, w_i)$ for $i \in \{0, 1\}$ and derive its equivalence with equality in \mathbb{V} .

Corollary 3.12. For $\bar{X}_0, \bar{X}_1 : \mathbb{V}$, we have

$$\mathbb{V}(\bar{X}_0, \bar{X}_1) \simeq \text{Eq}_{\mathbb{V}}(\bar{X}_0, \bar{X}_1). \quad \square$$

3.2.1 \mathbb{V} is n -truncated.

Theorem 3.13. Let (S, Pos) be an n -truncatedness retaining container. If $P(X)$ implies $\text{type}_{n-1}(X)$ then \mathbb{V} is n -truncated.

Proof. Given $\bar{X}_i \equiv (X_i, w_i) : \mathbb{V}$ for $i \in \{0, 1\}$ we proceed by induction on w_0 and w_1 to prove $\mathbb{V}(\bar{X}_0, \bar{X}_1)$ is $(n-1)$ -truncated. By Corollary 3.12 and the same reasoning as in the proof of Proposition 2.9, we have to concern ourselves only with the following pushout square³:

$$\begin{array}{ccc} \sum_{(p : \mathcal{U}^{\leq n}(X_0, X_1))} P(X_0) \times E(p) & \rightarrow & \sum_{(p : \mathcal{U}^{\leq n}(X_0, X_1))} P(X_0) \\ \downarrow & & \downarrow \\ \sum_{(p : \mathcal{U}^{\leq n}(X_0, X_1))} E(p) & \longrightarrow & \text{Eq}_{\mathbb{V}}(\bar{X}_0, \bar{X}_1) \end{array} \quad (5)$$

where $E(p) = \text{Eq}'_{\text{inV}}(p, (s_0, t_0), (s_1, t_1))$. By [24, Proposition A.11], it is enough to show the top right and bottom left corners are $(n-1)$ -truncated to conclude that $\text{Eq}_{\mathbb{V}}(\bar{X}_0, \bar{X}_1)$ is as well. $\sum_{(p : \mathcal{U}^{\leq n}(X_0, X_1))} P(X_0)$ is $(n-1)$ -truncated because $P(X_0)$ is a

³a variant of (3)

$$\text{Eq}'_{\text{inV}}(q, (s_0, t_0), (s_1, t_1)) \equiv_{\text{def}} \Sigma (q_s : s_0 =_{S(q)} s_1). \\ (e_t : \prod_{(Y: \mathcal{U}^{\leq n})} \prod_{(p_0, p_1)} p_0 =_{\text{Pos}(q, q_s, Y)} p_1 \rightarrow \text{Eq}_{\text{inV}}(\text{refl}, t_{0Y}(p_0), t_{1Y}(p_1)))$$

$$\text{Eq}_{\text{inV}}(q : X_0 = X_1, w_0, w_1) \equiv_{\text{def}} \begin{cases} P(X_0) \star \text{Eq}'_{\text{inV}}(q, c_0, c_1) & \text{if } w_0 \equiv \text{tcon}(c_0), w_1 \equiv \text{tcon}(c_1) \\ \text{by contractibility} & \text{if } w_0 \text{ or } w_1 \text{ given by ua}(\dots) \end{cases}$$

Figure 1. Definition of Eq_{inV} .

proposition and implies $\text{type}_{n-1}(X_0)$, so that $\mathcal{U}^{\leq n}(X_0, X_1)$ is $(n-1)$ -truncated by univalence. $\sum_{(p: \mathcal{U}^{\leq n}(X_0, X_1))} E(p)$ is equivalent to $(X_0, (s_0, t_0)) = (X_1, (s_1, t_1))$ by Proposition 3.11, so we can conclude its $(n-1)$ -truncatedness by using that (S, Pos) retains n -truncatedness, because its premise is satisfied by the induction hypothesis. \square

As the special case where P is constantly false, we obtain the folklore construction of 0-truncated “closed” universes.

Corollary 3.14. *Let (S, Pos) be a 0-truncatedness retaining container. If $P = \lambda X. \perp$ then V is 0-truncated.* \square

4 Models of n -truncated Type Theory with Univalence for $(n-1)$ -types.

In this section, we show that Martin-Löf type theory with function extensionality and the assumption that all types are n -truncated is consistent with univalence for $(n-1)$ -types.

Remark 4.1. The full strength of this statement is realized only with a sufficiently long chain of universes $\mathcal{U}_0, \dots, \mathcal{U}_k$, one included in the next. For if $k < n$, it is known [18, Section 6] how to modify a model of homotopy type theory (including univalence, but no higher inductive types) to be n -truncated by restricting types of “size” i (classified by \mathcal{U}_i) to i -types (and restricting all types to the n -truncated).

For the reason given in the above remark, we consider Martin-Löf type theory to come with an ω -indexed (cumulative) hierarchy of universes $\mathcal{U}_0, \mathcal{U}_1, \dots$. Alternatively, we could include higher inductive types, which in the presence of univalence for $(n-1)$ -types are still able to produce proper n -types.⁴ However, a key point is still for an n -truncated universe univalent for $(n-1)$ -types to be able to contain a code for a (smaller) universe of the same kind, and at this point we may as well consider a hierarchy of universes.

We use *categories with families* (cwfs) [12] as our notion of model of dependent type theory. They are models of a generalized algebraic theory [8] (as will all semantic notions considered here). Fixing the underlying category \mathcal{C} , we obtain a category of *cwf structures* on \mathcal{C} . We refer to a cwf structure on \mathcal{C} by its presheaf of types Ty , the presheaf Tm of terms on its category of elements $\int \text{Ty}$ left implicit.

⁴ For example, univalence for 0-types is sufficient to show that the circle S^1 is a proper 1-type. We consider an n -type *proper* if it is not an $(n-1)$ -type.

Let T stand for a choice of type formers, specified by a collection of rules that are generally natural in the context (one way to ensure this naturality is by demanding that these rules be interpretable in presheaves over the category of contexts and substitutions [5, 7]). Type formers can be standard type formers such as dependent sums, dependent products, or identity types, but also “axioms” such as function extensionality. As before, we have categories of *cwfs with type formers* T as well as *cwf structures with type formers* T on a fixed category \mathcal{C} .

Definition 4.2 ([4, Definition 2.4]). A *cwf hierarchy* Ty with *type formers* T on a category \mathcal{C} is a sequential diagram

$$\text{Ty}_0 \longrightarrow \text{Ty}_1 \longrightarrow \dots$$

of cwf structures with type formers T on \mathcal{C} .

Note that the *lifting maps* $\text{Ty}_i \rightarrow \text{Ty}_{i+1}$ preserve type formers T . As before, cwf hierarchies with type formers T assemble into a category.

Definition 4.3 ([4, Definition 2.5]). A *model of Martin-Löf type theory* with type formers T is a category \mathcal{C} with a cwf hierarchy Ty with type formers T on \mathcal{C} together with, for each i , a global section \mathcal{U}_i with an isomorphism $\text{El}_i : \text{Tm}_{i+1}(\Gamma, \mathcal{U}_i) \simeq \text{Ty}_i(\Gamma)$ natural in $\Gamma \in \mathcal{C}$.

In all uses of the above definition, we will implicitly assume that T contains at least dependent sums, dependent products, identity types, and finite coproducts. This makes available basic concepts of homotopy type theory such as being n -truncated (for an external number n). We write MLTT_T for the category such models, and $\text{MLTT}_T(\mathcal{C})$ if we wish to fix the underlying category. We say that \mathcal{C} is *n -truncated* (where $n \geq -2$) if all $A \in \text{Ty}_i(\Gamma)$ are n -truncated, naturally in $\Gamma \in \mathcal{C}$, for all i . As a type former (an axiom), we denote it $\text{Tr}(n)$.

We call \mathcal{U}_i the *i -th universe* of \mathcal{C} . Note that, in contrast to our internal reasoning, we explicitly reference the decoding natural transformation El_i . Given a generic property P of types (such as being n -truncated) forming an internal proposition, we say that \mathcal{C} satisfies *univalence for P* if the universe \mathcal{U}_i is P -univalent for all i in the sense of what follows Definition 2.2. We add a subscript $\text{UA}(n)$ to MLTT_T to indicate restriction to models with univalence for n -types.

Let T be a collection of type formers. Given $\mathcal{C} \in \text{MLTT}_T$ and $i \geq 0$, the type forming operations contained in T can be

encoded as internal operations on the universe \mathcal{U}_i . Assume that these internal operations restrict to the subuniverse $\mathcal{U}_i^{\leq n}$ of n -types. For well-behaved T , it is possible to find a global \mathcal{U}_i -indexed container C of size $i + 1$ such that for any family S over $\mathcal{U}_i^{\leq n}$, a lift of the given internal operations on $\mathcal{U}_i^{\leq n}$ to $\sum_{(X:\mathcal{U}_i^{\leq n})} S(X)$ corresponds to a C -algebra structure on S . Lastly, assume that C retains n -truncatedness in the sense of Definition 3.6. If all of this is the case, naturally in C , we say that T is n -benign.

Example 4.4. The basic type formers we implicitly require for a model of Martin-Löf type theory are n -benign for any $n \geq 0$. The associated $\mathcal{U}_i^{\leq n}$ -indexed containers are listed in Figure 2 (with the size index i omitted). Retention of n -truncatedness in the sense of Definition 3.6 follows the scheme of Example 3.7.

Example 4.5. Any type former that only has term forming operations is automatically n -benign. In the first place, this applies to axiom-style type formers such as function extensionality.

We are now ready to state the main result.

Theorem 4.6. *Let $n \geq 0$ and T be an n -benign choice of type formers, including function extensionality. Let (C, Ty) be a model of Martin-Löf type theory with type formers T that is univalent for $(n - 1)$ -types. Then there is an n -truncated model Ty' of Martin-Löf type theory with type formers T on C that is univalent for $(n - 1)$ -types. Furthermore, there is a morphism $\text{Ty}' \rightarrow \text{Ty}$ of cwf hierarchies with type formers T on C .*

Proof. Given a cwf structure Ty on a category C , note that a further cwf structure Ty' together with a morphism $\text{Ty}' \rightarrow \text{Ty}$ corresponds up to isomorphism to just a presheaf Ty' of types with a natural transformation $\text{Ty}' \rightarrow \text{Ty}$.⁵ The terms of Ty' are inherited (up to isomorphism) from those of Ty since terms correspond to sections of context projections and $\text{Ty}' \rightarrow \text{Ty}$ should preserve context extension. Abstractly speaking, the forgetful functor from cwf structures on C to discrete fibrations on C is itself a discrete fibration.

Let us further assume that Ty implements some type type formers T . To interpret T in Ty' such that $\text{Ty}' \rightarrow \text{Ty}$ preserves T , we only have to interpret the actual type forming operations of T in Ty' such that they are preserved by $\text{Ty}' \rightarrow \text{Ty}$; the term forming operations of T will then be uniquely inherited from Ty .

Let us now return to the situation of Theorem 4.6. The type forming operations of the type formers T in (C, Ty_i) can be encoded as internal operations on the universe \mathcal{U}_i .⁶ Since T is n -benign, these internal operations further restrict to the subuniverse $\mathcal{U}_i^{\leq n}$ of n -types. We now wish to define a

⁵This is immediate when switching from cwf's to the equivalent notion of categories with attributes.

⁶Note that this is only a bijective correspondence if we have the judgmental η -law for dependent products, but this is not required here.

global type V_i of size $i + 1$ with a map $V_i \rightarrow \mathcal{U}_i^{\leq n}$. Restricting El_i along this map, we can see V_i as a universe. Defining $\text{Ty}'_i(\Gamma) = \text{Tm}(\Gamma, V_i)$ with $\text{Ty}'_i \rightarrow \text{Ty}_i$ induced by $V_i \rightarrow \mathcal{U}_i^{\leq n}$, we then obtain the cwf structure Ty'_i with a map $\text{Ty}'_i \rightarrow \text{Ty}_i$. Interpreting T in Ty'_i compatible with Ty_i will follow from a (strict) lift of the internal type formation operations from $\mathcal{U}_i^{\leq n}$ to V_i . Finally, everything needs to be natural in $i \in \omega$.

Let us start with the base $i = 0$. We will define $V_0 \equiv_{\text{def}} \sum_{(X:\mathcal{U}_0^{\leq n})} \text{inV}_0$ for a family inV_0 over $\mathcal{U}_0^{\leq n}$, with $V_0 \rightarrow \mathcal{U}_0^{\leq n}$ the first projection. Using that T is n -benign, we have a $\mathcal{U}_0^{\leq n}$ -indexed container C_0 such that a lift of the internal formations operations from $\mathcal{U}_0^{\leq n}$ to V_0 corresponds to a C_0 -algebra structure on inV_0 . We now follow Section 3 for the construction of inV_0 from C_0 ; this means inV_0 is the type $_{n-1}$ -propositional indexed W-type $W_{C_0}^{\text{type}_{n-1}}$ as per [24, Subappendix B.5]. In particular, we obtain a C_0 -algebra structure on inV_0 . Note that V_0 is univalent for $(n - 1)$ -types by Lemma 3.9 and n -truncated by Theorem 3.13.

For general i , we let C'_i be the coproduct of the $\mathcal{U}_i^{\leq n}$ -indexed container C_i given from T being n -benign with the indexed container with shapes $0 \leq j < i$, with indexing of j being V_j (lifted to Ty_j), and no positions. We then define inV_i from C'_i as before. This guarantees that there are codes for the universes below i in V_i .

To make Ty' into a cwf hierarchy and $\text{Ty}' \rightarrow \text{Ty}$ into a morphism of cwf hierarchies, we need to construct, for every $i \geq 0$, a dotted morphism making the naturality square

$$\begin{array}{ccc} \text{Ty}'_i & \longrightarrow & \text{Ty}_i \\ \vdots & & \downarrow \\ \text{Ty}'_{i+1} & \longrightarrow & \text{Ty}_{i+1} \end{array}$$

of presheaves of types commute. This amounts to defining internal $V_i \rightarrow V_{i+1}$ making the square

$$\begin{array}{ccc} V_i & \longrightarrow & \mathcal{U}_i \\ \vdots & & \downarrow \text{lift} \\ V_{i+1} & \longrightarrow & \mathcal{U}_{i+1} \end{array}$$

commute strictly. In turns, this corresponds to an internal function

$$\text{inV}_i(X) \rightarrow \text{inV}_{i+1}(\text{lift}(X)) \quad (6)$$

for $X : \mathcal{U}_i^{\leq n}$. We define this by recursion for inV_i , noting that inV_{i+1} restricted along lift carries a C'_i -algebra structure, forgetting the code for the i -th universe in its C'_{i+1} -algebra structure.

It remains to check that the map $\text{Ty}'_i \rightarrow \text{Ty}'_{i+1}$ respects the type forming operations of T . This follows from (6) commuting strictly with C_i -algebra structures. This follows from the judgmental β -law of the higher inductive family inV_i .⁷

⁷This is the only place in our entire construction where judgmental β -laws for higher inductive types are needed. One might well regard it as an artifact of our universe hierarchy setup.

<ul style="list-style-type: none"> • Unit type: $S = 1,$ $t(\bullet) = 1,$ $Pos(A, x, y) = 0.$ 	<ul style="list-style-type: none"> • Dependent products: $S = \sum_{(A:\mathcal{U}^{\leq n})} A \rightarrow \mathcal{U}^{\leq n},$ $t(A, B) = \prod_{(a:A)} B(a),$ $Pos(A, B) = 1 + A,$ $s((A, B), \text{inl}(\bullet)) = A,$ $s((A, B), \text{inr}(a)) = B(a).$ 	<ul style="list-style-type: none"> • Empty type: $S = 1,$ $t(\bullet) = 0,$ $Pos(A, x, y) = 0.$
<ul style="list-style-type: none"> • Dependent sums: $S = \sum_{(A:\mathcal{U}^{\leq n})} A \rightarrow \mathcal{U}^{\leq n},$ $t(A, B) = \sum_{(a:A)} B(a),$ $Pos(A, B) = 1 + A,$ $s((A, B), \text{inl}(\bullet)) = A,$ $s((A, B), \text{inr}(a)) = B(a).$ 	<ul style="list-style-type: none"> • Identity types: $S = \sum_{(A:\mathcal{U}^{\leq n})} A \times A,$ $t(A, x, y) = A(x, y),$ $Pos(A, x, y) = 1,$ $s((A, x, y), \bullet) = A.$ 	<ul style="list-style-type: none"> • Binary coproducts: $S = \mathcal{U}^{\leq n} \times \mathcal{U}^{\leq n},$ $t(A, B) = A + B,$ $Pos(A, B) = 1 + 1,$ $s((A, B), \text{inl}(\bullet)) = A,$ $s((A, B), \text{inr}(\bullet)) = B.$

Figure 2. $\mathcal{U}^{\leq n}$ -indexed containers for basic type formers. The specifying data is given in a slightly alternate form: a type of shapes S , a target function $t : S \rightarrow \mathcal{U}_i^{\leq n}$, a family of positions Pos over S , and a source function $s : \prod_{s:S} Pos(s) \rightarrow \mathcal{U}_i^{\leq n}$. This corresponds to a polynomial functor $\mathcal{U}^{\leq n} \leftarrow S \rightarrow P \rightarrow \mathcal{U}^{\leq n}$ with middle arrow a fibration. The actual indexed container is obtained by taking fibers using the identity type.

It remains to check that Ty' has universes as required by Definition 4.3. Indeed, the i -th universe \mathcal{U}_i' is simply given by V_i itself, with El_i' the identity isomorphism. \square

Corollary 4.7. *Relative to Martin-Löf type theory with function extensionality and univalence for $(n - 1)$ -types (and any further n -benign type formers), if the addition of pushouts and propositionally truncated indexed W -types is consistent, then it is consistent to assume that all types are n -truncated.*

Proof. Given a model for the former theory, we obtain a model (on the same category) of the latter theory by Theorem 4.6. By construction, the empty type is inhabited in this model exactly if it is inhabited in the old model. \square

Corollary 4.8. *In Martin-Löf type theory with function extensionality and univalence for $(n - 1)$ -types (and any further n -benign type formers implemented by a known model of homotopy type theory), it is consistent to assume that all types are n -truncated.*

Proof. Apply Corollary 4.7 to a model of homotopy type theory such as simplicial sets or cubical sets that supports higher inductive families. \square

5 A Cubical Type Theory with UIP, Propositional Extensionality, and Homotopy Canonicity

In [11] the authors establish the homotopy canonicity property for a cubical type theory without judgmental equations for the box filling operations. Here we will follow that proof to prove homotopy canonicity for a cubical type theory with an axiomatic UIP principle and propositional extensionality

given by a modified Glue-type. To keep this section brief, we closely follow their notation.

5.1 0-truncated Cubical Cwf

We take the definition of cubical cwf from [11] and adapt it by adding a new trunc operation and an extra argument to Glue-types. The definition is internal to the category of cubical sets of [9]. A minor difference to [11], following the previous section, we only require a sequential diagram of Ty_i presheaves, without topmost Ty , and we do not require the lifting map $\text{Ty}_i \rightarrow \text{Ty}_{i+1}$ to be mono.

Given $A : \text{Ty}_i(\Gamma)$, we define $\text{isProp}(A)$ and $\text{isSet}(A)$ in $\text{Ty}_i(\Gamma)$ using the Path-type former.

- **Glue types.** Given $A : \text{Ty}_i(\Gamma)$, $A_p : \text{Elem}(\Gamma, \text{isProp}(A))$, $\varphi : \mathbb{F}$, $T : [\varphi] \rightarrow \text{Ty}_i(\Gamma)$, and $e : \text{Elem}(\Gamma, \text{Equiv}(T \text{ tt}, A))$, we have the *glueing* $\text{Glue}(A, A_p, \varphi, T, e)$ in $\text{Ty}_i(\Gamma)$, equal to $T \text{ tt}$ on φ . We also have $\text{glue}(a, t)$, unglue , and their equations as described in [11, Sec. 1.3].
- **0-truncation operation.** Given A in $\text{Ty}_i(\Gamma)$ we have $\text{trunc}(A)$ in $\text{Elem}(\Gamma, \text{isSet}(A))$. No equations are needed other than stability under substitution.

5.2 Standard Model

We now work in the category of cubical sets of [9]. It satisfies the assumptions listed at the top of [11, Section 2.2], so we have a hierarchy of universes of fibrant types $\mathcal{U}_i^{\text{fib}}$, defined from a cumulative hierarchy of universes of presheaves \mathcal{U}_i for $i \in \{0, 1, \dots, \omega\}$. Using it as a model of cubical type theory with uniformly indexed higher inductive types, we replay the construction from Section 3 and obtain a family

$\text{inV} : (\mathcal{U}_i^{\text{fib}})^{\leq 0} \rightarrow \mathcal{U}_{i+1}^{\text{fib}}$. Then we take

$$\mathbb{V}_i \equiv \sum_{(A : (\mathcal{U}_i^{\text{fib}})^{\leq 0})} \text{inV}(A) : \mathcal{U}_{i+1}^{\text{fib}}$$

with $\text{El}_{\mathbb{V}_i} : \mathbb{V} \rightarrow (\mathcal{U}_i^{\text{fib}})^{\leq 0}$.

Just as [11, Section 2.3] defines the standard model as an internal cwf from the universes $\mathcal{U}_i^{\text{fib}}$, we define the standard model from the universes \mathbb{V}_i :

- Con is the category with objects in \mathcal{U}_ω and functions between them as morphism,
- the types of size i over $\Gamma : \mathcal{U}_\omega$ are maps $\Gamma \rightarrow \mathbb{V}_i$,
- the elements of $A : \Gamma \rightarrow \mathbb{V}_i$ are $\Pi(\rho : \Gamma). \text{El}_{\mathbb{V}_i}(A \rho)$.

We will often omit the use of $\text{El}_{\mathbb{V}_i}$ to lighten the notational burden.

Type formers $\Pi, \Sigma, \mathbb{N}, \text{Path}$ and universes are given by including a code for them in the container (S, Pos) for \mathbb{V}_i . The filling operation is derived from the one for $\mathcal{U}_i^{\text{fib}}$. Glue-types are handled below.

5.2.1 A Code for Glue in \mathbb{V}_i . One would think that inV might need an explicit constructor for Glue. However, the path constructor ua of inV suffices to derive one, given Glue for $\mathcal{U}_i^{\text{fib}}$ and fibrancy of inV .

Given $\Gamma : \mathcal{U}_\omega, A : \Gamma \rightarrow \mathbb{V}_i, A_p : \Pi(\rho : \Gamma). \text{isProp}(A \rho), \varphi : \mathbb{F}, T : [\varphi] \rightarrow \Gamma \rightarrow \mathbb{V}_i$ and $e : [\varphi] \rightarrow \text{Equiv}(T \text{tt}, A)$, we wish to define $\text{Glue}(A, A_p, \varphi, T, e) : \Gamma \rightarrow \mathbb{V}_i$. We take

$$\text{Glue}(A, A_p, \varphi, T, e) \rho \equiv_{\text{def}} (G, w_G)$$

where $G = \text{Glue}(\text{El}_{\mathbb{V}_i}(A \rho), \varphi, \text{El}_{\mathbb{V}_i} \circ (T \rho), e \rho) : \mathcal{U}_i^{\text{fib}}$, and given $w_A \equiv A \rho.2$ and $w_T \equiv \lambda o. T \circ \rho.2$, we obtain w_{Glue} by first transporting $w_A : \text{inV}(\text{El}_{\mathbb{V}_i}(A \rho))$ to $w'_G : \text{inV}(G)$ by the canonical path between the two indices, and then composing under $[\varphi]$ with a path between w'_G and w_T built by ua . The latter is possible because, assuming $[\varphi]$, both w'_G and w_T are codes for $\text{El}_{\mathbb{V}_i}(T \text{tt} \rho)$, which is propositional by $A_p \rho$. Note that with this correction $\text{Glue}(A, A_p, \varphi, T, e) \equiv T \text{tt}$ when $[\varphi] \equiv \top$. One then checks that the code so defined commutes with the lifting maps $\mathbb{V}_i \rightarrow \mathbb{V}_{i+1}$.

5.3 Scoring Model

Given a cubical cwf \mathcal{M} (denoted by $\text{Con}, \text{Ty}_i, \text{Elem}, \dots$), we want to define a new cubical cwf \mathcal{M}^* , (denoted by $\text{Con}^*, \text{Ty}_i^*, \text{Elem}^*, \dots$) as the Artting glueing of \mathcal{M} along an internal global sections functor $|-|$. We assume \mathcal{M} size-compatible with the universes \mathcal{U}_i as in [11, Sec. 3]. In [11], the functor $|-|$ targets the standard model directly, given that $\text{Elem}(1, A)$ is a fibrant type. In our case, we have to include a code for it in inV_i , as in extending the container (S, Pos) , as follows:

- given $A : \text{Ty}_i(1)$, a constructor $[A] : \text{inV}_i(\text{Elem}(1, A))$.

Note that both $\text{Ty}_i(1)$ and $\text{Elem}(1, A)$ are 0-truncated, because the trunc operation implies $\text{isSet}(\text{Elem}(1, A))$ for any A in $\text{Ty}_j(1)$, and $\text{Ty}_i(1)$ itself is equivalent to $\text{Elem}(1, \mathcal{U}_i)$. This makes sure that the extended container still preserves 0-truncatedness.

For the definition of natural numbers in \mathcal{M}^* , we will also need a code for the type family $\mathbb{N}' : \text{Elem}(1, \mathbb{N}) \rightarrow \mathcal{U}_0^{\text{fib}}$ defined in [11, Appendix B]:

- given $n : \text{Elem}(1, \mathbb{N})$, a constructor $\mathbb{N}'(n) : \text{inV}_0(\mathbb{N}' n)$

where $\mathbb{N}' n$ can be shown to be 0-truncated by Corollary 3.14.

The functor $|-|$ is then given on contexts, types, and elements of \mathcal{M} like so:

- $|\Gamma| \equiv_{\text{def}} \text{Hom}_{\mathcal{M}}(1, \Gamma)$,
- $|A| \rho \equiv_{\text{def}} (\text{Elem}(1, A), [A \rho])$,
- $|a| \rho \equiv_{\text{def}} a \rho$.

We now define the scoring model \mathcal{M}^* , starting with the cwf components.

- A context $(\Gamma, \Gamma') : \text{Con}^*$ consists of $\Gamma : \text{Con}$ in \mathcal{M} and a family $\Gamma' : |\Gamma| \rightarrow \mathcal{U}_\omega$.
- A type $(A, A') : \text{Ty}_i^*(\Gamma, \Gamma')$ consists of a type $A : \text{Ty}_i(\Gamma)$ in \mathcal{M} and a family

$$A' : \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho) \rightarrow |A| \rho \rightarrow \mathbb{V}_i$$

of proof-relevant predicates over it.

- An element $(a, a') : \text{Elem}^*((\Gamma, \Gamma'), (A, A'))$ consists of an element $a : \text{Elem}(\Gamma, A)$ in \mathcal{M} and

$$a' : \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho) \rightarrow A(\rho, \rho', a \rho).$$

We observe that this definitions differs from the one given in Coquand et al. [11] only by the use of \mathbb{V}_i in place of $\mathcal{U}_i^{\text{fib}}$ to define Ty_i^* . As such we will not repeat here the details about the rest of the cwf structure or the shared type formers and operations, and instead discuss only Glue^* and tr^* .

5.3.1 Glue-types.

Lemma 5.1. *Let (A, A') in $\text{Ty}_i^*(\Gamma, \Gamma')$. The following statements are logically equivalent, naturally in (Γ, Γ') :*

$$\text{Elem}^*((\Gamma, \Gamma'), \text{isProp}^*(A, A')) \quad (7)$$

$$\begin{aligned} & \text{Elem}(\Gamma, \text{isProp}(A)) \\ & \times \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho). \text{isProp}(\Sigma(a : |A| \rho). A'(\rho, \rho' a)) \quad (8) \end{aligned}$$

$$\begin{aligned} & \text{Elem}(\Gamma, \text{isProp}(A)) \\ & \times \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho)(a : |A| \rho). \text{isProp}(A'(\rho, \rho' a)) \quad (9) \end{aligned}$$

Proof. Given (7), we have $A_p : \text{Elem}(\Gamma, \text{isProp}(A))$ and a proof that one can fill lines in A' over lines produced by $|A_p|$; that is enough to fill lines in the Σ -type in (8). From there, we derive (9): since $|A| \rho$ is propositional, any path from a to a is constant. Going back to (7) requires only to contract a path in $|A| \rho$. \square

Let (A, A') in $\text{Ty}_i^*(\Gamma, \Gamma')$, φ in \mathbb{F} , $\langle T, T' \rangle$ in $[\varphi] \rightarrow \text{Ty}_i^*(\Gamma, \Gamma')$, $\langle e, e' \rangle$ in $\text{Elem}^*((\Gamma, \Gamma'), \text{Equiv}^*((T \text{tt}, T' \text{tt}), (A, A')))$, and (A_p, A'_p) in $\text{Elem}^*((\Gamma, \Gamma'), \text{isProp}^*(A, A'))$. We follow the recipe of [11, Sec. 3.2.6] and define

$$\text{Glue}^*((A, A'), (A_p, A'_p), \varphi, \langle T, T' \rangle, \langle e, e' \rangle)$$

as $(\text{Glue}(A, A_p, \varphi, T, e), G')$ where $G' \rho \rho' (glue(a, t))$ is defined as the Glue-type in \mathbb{V}_i of $A' \rho \rho' a$ and $T' \text{tt} \rho \rho' (t \text{tt})$ along φ . In our case we also have to provide a proof of

isProp($A' \rho \rho' a$), which we obtain from (A_p, A'_p) by applying Lemma 5.1, going from (7) to (9).

5.3.2 trunc-operation.

Lemma 5.2. *Let $(A, A') : \text{Ty}_i^*(\Gamma, \Gamma')$. The following statements are logically equivalent, naturally in (Γ, Γ') :*

$$\text{Elem}^*((\Gamma, \Gamma'), \text{isSet}^*(A, A')) \quad (10)$$

$$\begin{aligned} & \text{Elem}(\Gamma, \text{isSet}(A)) \\ & \times \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho). \text{isSet}(\Sigma(a : |A| \rho). A'(\rho, \rho' a)) \end{aligned} \quad (11)$$

$$\begin{aligned} & \text{Elem}(\Gamma, \text{isSet}(A)) \\ & \times \Pi(\rho : |\Gamma|)(\rho' : \Gamma' \rho)(a : |A| \rho). \text{isSet}(A'(\rho, \rho' a)) \end{aligned} \quad (12)$$

Proof. This follows the same strategy as Lemma 5.1, except this time filling and contracting squares rather than lines. \square

Let $(A, A') : \text{Ty}_i^*(\Gamma, \Gamma')$. We define

$$\text{trunc}^*(A, A') : \text{Elem}^*((\Gamma, \Gamma'), \text{isSet}^*(A, A))$$

by applying Lemma 5.2 in the direction from (10) to (12). to the pair of $\text{trunc}(A)$ and trunc' where

$$\text{trunc}' \rho \rho' a : \text{isSet}(\text{El}_{V_i}(A'(\rho, \rho' a)))$$

is given by $\text{El}_{V_i}(A'(\rho, \rho' a)) : (\text{U}_i^{\text{fib}})^{\leq 0}$.

Given the above constructions, one mechanically verifies the necessary laws to obtain the following statement.

Theorem 5.3 (Scoring). *Given any 0-truncated cubical cwf \mathcal{M} that is size-compatible in the sense of [11, Sec. 3], the scoring \mathcal{M}^* is a 0-truncated cubical cwf with operations defined as above. We further have a morphism $\mathcal{M}^* \rightarrow \mathcal{M}$ of 0-truncated cubical cwf's given by the first projection.* \square

We state homotopy canonicity with reference to the initial 0-truncated cubical cwf \mathcal{I} , whose existence and size-compatibility is justified as in [11, Sec. 4]. The proof of the theorem also follows the argument given in that section.

Theorem 5.4 (Homotopy canonicity). *In the internal language of the cubical sets category of [9], given a closed natural $n : \text{Elem}(1, \mathbb{N})$ in the initial model \mathcal{I} , we have a numeral $k : \mathbb{N}$ with $p : \text{Elem}(1, \text{Path}(\mathbb{N}, n, S^k(0)))$.* \square

6 Related Work and Conclusion

6.1 Related Work

In the realm of type theories with UIP and function extensionality, XTT [25] is a non-univalent variant of CTT that takes the extra step of making UIP hold judgmentally, in the spirit of observational type theory (OTT) [3]. As formulated XTT does not provide propositional extensionality and requires a typecase operation within the theory for (strict) canonicity. OTT does include propositional extensionality, but only for a universe of propositions closed under a specific set of type formers that made it possible to assume judgmental proof irrelevance for such propositions. We conjecture that by introducing UIP (or n -truncatedness) only as a path

equality we will be able to refine our theory to one with strict canonicity without encountering similar limitations. Regarding strict propositions, i.e. where any two elements are strictly equal in the model, the semantics for a univalent universe of them within the cubical sets model is described in [10]. However the corresponding universe of strict sets is not a strict set itself. Such semantics are used in [14] to justify the addition of a primitive universe of strict propositions sProp.

6.2 Conclusion

We proved consistency for a theory with n -truncatedness and univalence for $(n - 1)$ -types. We also showed homotopy canonicity for cubical variant of such a theory. The main technical tool used was an n -truncated universe of n -types that is also univalent for $(n - 1)$ -types. We would like to stress that such a universe can also be used directly in HoTT with indexed higher inductive types, for applications that do not mind the universe being limited to a fixed set of type formers.

Acknowledgments

Christian Sattler was supported by USAF grant FA9550-16-1-0029. Andrea Vezzosi was supported by a research grant (13156) from VILLUM FONDEN.

References

- [1] Thorsten Altenkirch, Neil Ghani, Peter Hancock, Conor McBride, and Peter Morris. 2015. Indexed containers. *Journal of Functional Programming* 25 (2015), e5. <https://doi.org/10.1017/S095679681500009X>
- [2] Thorsten Altenkirch and Ambrus Kaposi. 2016. Type Theory in Type Theory Using Quotient Inductive Types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (St. Petersburg, FL, USA) (POPL '16). ACM, New York, NY, USA, 18–29. <https://doi.org/10.1145/2837614.2837638>
- [3] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. 2007. Observational Equality, Now!. In *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification* (Freiburg, Germany) (PLPV '07). Association for Computing Machinery, New York, NY, USA, 57–68. <https://doi.org/10.1145/1292597.1292608>
- [4] Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. 2019. Two-Level Type Theory and Applications. *arXiv preprint arXiv:1801.01568v3* (2019).
- [5] Steve Awodey. 2018. Natural models of homotopy type theory. *Mathematical Structures in Computer Science* 28, 2 (2018), 241–286.
- [6] Steve Awodey and Michael A. Warren. 2009. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society* 146, 1 (2009), 45–55. <https://doi.org/10.1017/S0305004108001783>
- [7] Paolo Capriotti. 2016. *Models of Type Theory with Strict Equality*. Ph.D. Dissertation. School of Computer Science, University of Nottingham.
- [8] John Cartmell. 1986. Generalised algebraic theories and contextual categories. *Annals of pure and applied logic* 32 (1986), 209–243.
- [9] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *Types for Proofs and Programs (TYPES 2015) (LIPIcs)*, Vol. 69. 5:1–5:34.
- [10] Thierry Coquand. 2016. Universe of Bishop sets.(2016). www.cse.chalmers.se/~coquand/bishop.pdf

- [11] Thierry Coquand, Simon Huber, and Christian Sattler. 2019. Homotopy Canonicity for Cubical Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Herman Geuvers (Ed.), Vol. 131. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 11:1–11:23. <https://doi.org/10.4230/LIPIcs.FSCD.2019.11>
- [12] Peter Dybjer. 1995. Internal type theory. In *Types for Proofs and Programs (TYPES) (Lecture Notes in Computer Science)*, Stefano Berardi and Mario Coppo (Eds.), Vol. 1158. Springer-Verlag, 120–134. https://doi.org/10.1007/3-540-61780-9_66
- [13] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. 2018. Finite Sets in Homotopy Type Theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (Los Angeles, CA, USA) (CPP 2018)*. Association for Computing Machinery, New York, NY, USA, 2018–214. <https://doi.org/10.1145/3167085>
- [14] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. 2019. Definitional Proof-Irrelevance without K. *Proc. ACM Program. Lang.* 3, POPL, Article 3 (Jan. 2019), 28 pages. <https://doi.org/10.1145/3290316>
- [15] Martin Hofmann and Thomas Streicher. 1998. The groupoid interpretation of type theory. *Twenty-five years of constructive type theory (Venice, 1995)* 36 (1998), 83–111.
- [16] Ambrus Kaposi and András Kovács. 2018. A syntax for higher inductive-inductive types. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [17] Nicolai Kraus. 2014. The General Universal Property of the Propositional Truncation. In *20th International Conference on Types for Proofs and Programs, TYPES 2014, May 12-15, 2014, Paris, France*. 111–145. <https://doi.org/10.4230/LIPIcs.TYPES.2014.111>
- [18] Nicolai Kraus and Christian Sattler. 2015. Higher Homotopies in a Hierarchy of Univalent Universes. *ACM Transactions on Computational Logic (TOCL)* 16, 2 (2015), 18.
- [19] Peter LeFanu Lumsdaine and Michael Shulman. 2017. Semantics of higher inductive types. In *Mathematical Proceedings of the Cambridge Philosophical Society*. Cambridge University Press, 1–50.
- [20] Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73*, H.E. Rose and J.C. Shepherdson (Eds.). Studies in Logic and the Foundations of Mathematics, Vol. 80. Elsevier, 73 – 118. [https://doi.org/10.1016/S0049-237X\(08\)71945-1](https://doi.org/10.1016/S0049-237X(08)71945-1)
- [21] Fredrik Nordvall Forsberg, Chuangjie Xu, and Neil Ghani. 2020. Three Equivalent Ordinal Notation Systems in Cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. <https://doi.org/10.1145/3372885.3373835>
- [22] Egbert Rijke. 2018. Introduction to Homotopy Type Theory. Lecture notes.
- [23] Egbert Rijke, Michael Shulman, and Bas Spitters. 2017. Modalities in homotopy type theory. *arXiv preprint arXiv:1706.07526* (2017).
- [24] Christian Sattler and Andrea Vezzosi. 2020. Partial Univalence in n-truncated Type Theory. (2020). [arXiv:2005.00260v1](https://arxiv.org/abs/2005.00260v1) [cs.LO]
- [25] Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. 2019. Cubical Syntax for Reflection-Free Extensional Equality. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Herman Geuvers (Ed.), Vol. 131. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 31:1–31:25. <https://doi.org/10.4230/LIPIcs.FSCD.2019.31>
- [26] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study.
- [27] Niccolò Veltri and Andrea Vezzosi. 2020. Formalizing π -calculus in Guarded Cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. <https://doi.org/10.1145/3372885.3373814>