parametricity-based formalization of semi-simplicial and semi-cubical sets

Hugo Herbelin
Université P. ris-Cité, Inri, CNRS, IRIF, P. ris
Hugo.Herbelin@inri.fr

Abstract
Semi simplicial and semi-cubical sets are commonly defined as presheaves over respectively, the semi simplex or semi-cube category. Homotopy Type Theory then populates internal tivity definition, where the set of n-simplices or n-cubes re labeled as groups into the fibers over the face spaces, defining the corresponding to ceteris. We know that semi-simplicial and semi-cubical sets are related to iterated Reynolds p r metricity, respectively in its un indexed form. We exploit this correspondence to develop an uniform indexed definition of both augmented semi-simplicial and semi-cubical sets, fully formalize it in Coq.

1 Introduction

Fibered vs indexed presentation of semi-simplicial and semi-cubical sets

Family of sets can commonly be represented in two ways: as a family properly speaking, indexed by the elements of a given set S, or as a set T together with a map from T to S, which specifies for each element of T its dependency on S. In the former case, we call it an indexed presentation. In the latter case, the set associated to a given element of S is the fiber of this element, so we call it a fibered presentation. The two presentations are equivalent and the equivalence can be phrased concisely in the language of homotopy type theory [The13] as the fibered/indexed equivalence:

\[(\text{fibered}) \quad \Sigma T \quad \text{HSet.} \quad (T \to S)) \simeq \quad S \to \text{HSet} \quad (\text{indexed})\]

Here, HSet represents in homotopy type theory the subset of types within a given universe where equality of any two elements has at most one proof.

Presheaf on an category a family of sets indexed by the object of the category with maps indexed by the morphisms. s such, it lives on the indexed side of the equivalence, contrasting with the fibered side, where we have discrete Grothendieck fibrations [LR20]. However, there are situations where a presheaf can also be seen as living on the fibered side of the equivalence. This happens when the indexing category is direct, or has a downwards-well-founded collection of non-identity morphisms. Let us consider, for instance, the case of a semi-cubical set [GM03, BM17] presented with 2n face maps from the set of n-cubes to the set of n - 1)-cubes. Formulated in type theory, the corresponding presheaf definition of a semi-cubical set prescribes a family of sets and face maps between them as follows:

\[X_0 \quad \text{HSet} \quad \begin{cases} \partial^R \leftrightarrow \partial^L \leftrightarrow \partial^S \leftrightarrow X_1 \\ \partial^R \leftrightarrow \partial^L \leftrightarrow \partial^S \leftrightarrow X_2 \quad \text{HSet} \quad \ldots \end{cases}\]

In an informal discussion, alternative nomenclatures were proposed: fibration/family equivalence and unbundled/bundled equivalence. The fibered/indexed nomenclature echoes the Grothendieck construction of fibered categories from indexed categories. The most elementary instance of the equivalence, with Type instead of HSet, is sometimes called “Grothendieck construction for dummies”, and its proof requires univalence [The13].
up to cubical faces identities. Here, $X_1$ can be seen as a family over $X_0 \times X_0$, and $X_2$ can be seen as a family over $X_1 \times X_1 \times X_1 \times X_1$, in the fibered presentation. This suggests an alternative indexed presentation of the presheaf as a stratified sequence of families indexed by families of lower rank, taking into account the coherence conditions to prevent duplications. Formulated in type theory, it takes the form:

$$
egin{align*}
X_0 & \vdash X_0 \\
X_1 & \vdash X_0 \times X_0 \\
X_2 (a, b) & \vdash (X_1 a, b) \times X_1 (c, d) \\
& \vdash (X_1 a, c) \times X_1 (b, d) \\
\vdots
\end{align*}
$$

The idea for such an indexed presentation of presheaves over a direct category was mentioned at the Univalent Foundations year in the context of defining semi-simplicial types\(^b\), few constructions have been proposed since then. The first construction by [Voe12] relies on the presentation of semi-simplicial sets as a presheaf over increasing injective maps between finite ordinals. The second, by [Her15]\(^c\) formalized in the Coq proof assistant, relies on the presentation of semi-simplicial sets as a presheaf over face maps. Another by [PL15] formalized in an emulation of logic-enriched homotopy type theory in the Plastic proof assistant, and yet another by [CK17] formalized in an emulation of a two-level type theory in the gda proof assistant\(^d\), rely on the presentation of the semi-simplicial category from increasing injective functions over finite ordinals.

The indexed definition of a presheaf over a direct category is technically more involved than the presheaf definition, as it requires hard-wiring in the structure the dependencies between elements of the sets of the presheaf, including the coherence conditions between these dependencies, such as taking the $i$-th face of the $j$-th face of a $n$-simplex being the same as taking the $j-1$-th face of the $i$-th face (when $j > i$). However, exhibiting a concrete instance of a presheaf in indexed form only requires providing the families, since the responsibility of defining maps and showing the coherence conditions is already accounted for in the definition of the structure.

**Reynolds parametricity and its unary and binary variants**

In the context of functional programming, Reynolds parametricity ([Rey83]) interprets types as relations characterizing the observational behavior of programs of this type. More generally, families over a product of sets, or correspondences, can be used in place of relations. Parametricity can then be iterated, and relying on the fibered presentation of correspondences as spans, it has been noted that iterated Reynolds parametricity has the structure of a cubical set [K15, Mou16, JS17, Moe21, Moe22]. We obtain a unary variant of Reynolds binary parametricity by using predicates or families instead of relations or correspondences, and this is a form of realizability [BM12, La14, Mou16]. It has then been noted that iterated unary parametricity has the structure of an augmented simplicial set\(^e\). This suggests that the definition of augmented semi-simplicial sets and semi-cubical sets can both be seen as instances of a more general construction, which we call $\nu$-sets, of presheaves over a $\nu$-semi-shape category made of words of some cardinal $\nu + 1$, where $\nu = 1$ gives augmented semi-simplicial sets and $\nu = 2$ gives semi-cubical sets.

**Contribution**

The main contribution of the paper is to describe the details of a recipe that uniformly characterizes unary and binary iterated parametricity in indexed form, and to derive from it a new indexed presentation, called indexed $\nu$-sets, of augmented semi-simplicial and semi-cubical sets.

Our work is a step in the direction of the program initiated in [K15] to develop parametricity-based models of parametric type theory [BCM15, NVD17, CH20] and cubical type theory [BCH13, CCHM18, BC+21], which are closer to the syntax of type theory, and are likely to better reflect the definitional properties of type theory than presheaf-based cubical sets would. For example, consider the loss of definitional properties when interpreting “indexed” dependent types of type theory as “fibrations” in models such as locally cartesian closed categories [CGH14].

Our mechanization can be found at github.com/autagnon/bonak. The construction was conceived in Summer 2019, and the mechanization began in late 2019. A sketch of the construction was presented at the 2020 HoTT-UF workshop, and the completion of the mechanization was reported at the TYPES 2022 conference.

\(^b\)ncatlab.org/mlab/show/semi-simplicial+types+in+homotopy+type+theory

\(^c\)In hindsight, the title of the paper “dependently-typed construction of semi-simplicial types” is somewhat confusing: it implicitly claimed to construct semi-simplicial types, but the construction was done in a type theory with Uniqueness of Identity Proofs. Consequentially, what was really obtained was an indexed presentation of semi-simplicial sets. The confusion was however, common at the time.

\(^d\)github.com/nicolalkraus/HoTT gda/blob/master/nicolai/SemiSimp/SSTypes.agda

\(^e\)Private communication with Hugo Moeneclaey and Thorsten Iltenkirch.
II Semi-simplicial and semi-cubical sets

In this section, we generalize semi-simplicial and semi-cubical sets to \(\nu\)-sets, subsuming the earlier definitions. We start with some introductory material on semi-simplicial and semi-cubical sets.

augmented semi-simplicial sets

Augmented semi-simplicial sets are defined similarly to semi-simplicial sets, except that the connected components are additionally dependent on a “color”. Conversely, semi-simplicial sets can be seen as augmented semi-simplicial sets over the singleton set of a fixed color. Let us associate dimension 0 to colors; then, points are dimension 1, lines are dimension 2, and so on. There is hence a shift by one when compared to semi-simplicial sets.

While ordinary semi-simplicial sets are presheaves over the semi-simplex category, augmented semi-simplicial sets are presheaves over \(\nu\). There are different ways to define \(\nu\), up to equivalence, and we use a definition that can be extended to semi-cubical sets in a straightforward manner.

II.0.1 Notation [Finite sequences] We denote finite sequences by \(i_1 \ldots i_n\) for \(i_j\) ranging over some domain. The empty sequence is written \(\emptyset\).

II.0.2 Definition \([\nu]\) The definition of \(\nu\) is shown below. Note that, if \(g \circ f\) is well-defined, then the length of \(f\) is less than that of \(g\). It can be shown that composition is associative and that \(id\) is neutral.

\[
\begin{align*}
\text{Obj}_\nu &= \{ l \in \{0, *\}^n \mid \text{number of } * \text{ in } l = p \} \\
\text{Hom}_\nu(p, n) &= \{ f : l \in \{0, *\}^n \mid \text{number of } * \text{ in } l = p \} \\
g \circ f &= \begin{cases} f & \text{if } g = \emptyset \\
0 \circ f & \text{if } g = 0 g' \\
0 g' \circ f' & \text{if } g = g', f = a f', \text{where } a = 0 \text{ or } * \\
\text{id} = \star \ldots \star n \text{ times for id } \in \text{Hom}_\nu(n, n) \end{cases}
\end{align*}
\]

II.0.3 Definition [Set \(\nu\)] We define the category of augmented semi-simplicial sets as the functor category:

\[
\text{Set}_\nu = \text{Set}^{\text{op}}
\]

To provide examples, we define the standard augmented \(n\)-semi-simplex.

II.0.4 Definition \([n_\nu]\) The standard augmented \(n\)-semi-simplex \(n_\nu\) is defined as the Yoneda embedding of \(n \in \text{Obj}_\nu\):

\[
\begin{align*}
n_\nu \colon \text{Set}_\nu &\to \text{Set}^{\text{op}} \\
\text{Set}_\nu(p) &\mapsto \text{Hom}_\nu(p, n) \\
n_\nu(f) &\mapsto \lambda g. g \circ f
\end{align*}
\]

The standard augmented 0-semi-simplex is a singleton made of one color (in this case, black). Standard augmented \(n\)-semi-simplices for \(n \geq 1\) have a geometric interpretation, and we illustrate them for dimensions 1, 2, and 3.

II.0.5 Example \([1_\nu]\) The standard augmented 1-semi-simplex can be pictured as a point, colored black, corresponding to the unique morphism in \(\text{Hom} \ 0, 1\). This point is the identity in \(\text{Hom} \ 1, 1\); it is hence shown as a singleton \(*\).

\[
\begin{array}{c}
\ast
\end{array}
\]

II.0.6 Example \([2_\nu]\) The standard augmented 2-semi-simplex is drawn as two points, given by \(\text{Hom} \ 1, 2\), along with a line connecting them, given by \(\text{Hom} \ 2, 2\). We use black to denote the unique morphisms in \(\text{Hom} \ 0, 1\) and \(\text{Hom} \ 0, 2\).

\[
\begin{array}{ccc}
\ast 0 & \ast 0 & \ast
\end{array}
\]
II.0.7 Example [ ] \( \frac{3}{3} \) is drawn as three points, given by \( \text{Hom } 1,3 \), three lines connecting them, given by \( \text{Hom } 2,3 \), and a triangular filler given by \( \text{Hom } 3,3 \).

![Diagram](image)

More generally, the standard augmented \( n+1 \)-semi-simplex can be obtained by taking a copy of the standard augmented \( n \)-semi-simplex serving as a base, and gluing on top of it another copy lifted by one dimension. In the second copy, the color becomes an extra point, the points become lines connecting the points of the base to the extra point, and so on. In particular, the components of the base are those of the standard augmented \( n \)-semi-simplex postfixed by \( 0 \) while the components of the lifted copy are postfixed by \( * \). Note that the components may be oriented by letting each \( n \)-dimensional component point to the \( n+1 \)-dimensional component obtained by replacing the leftmost \( * \) of the \( n \)-dimensional component with \( 0 \).

**Semi-cubical sets**

Semi-cubical sets are defined like augmented semi-simplicial sets except that \( + \) is replaced by \( \circ \) in which we take sequences of \( L, R \), and \( \ast \), instead of sequences of \( 0 \) and \( * \).

II.0.8 Definition [ ] The definition of \( \circ \) is shown below. The symbols \( L \) and \( R \) indicate opposite faces of a cube.

\[
\text{Obj} = \text{Hom} \ (p,n) = \{ l \in \{L,R,\ast\}^n \mid \text{number of } \ast \text{ in } l = p \}
\]

\[
g \circ f = \begin{cases} 
  f & \text{if } g - \\
  a \ g' \circ f & \text{if } g = a g', \text{where } a = L \text{ or } R \\
  a \ g' \circ f' & \text{if } g - \ast g', f = a f', \text{where } a = L, R, \text{ or } \ast
\end{cases}
\]

\[
\text{id} = \ast \ldots \ast \ n \text{ times}
\]

Gain, if \( g \circ f \) is well-defined, then the length of \( f \) is less than that of \( g \). It can be shown that composition is associative and that \( \text{id} \) is neutral.

II.0.9 Definition [Set ] We define the category of semi-cubical sets as the functor category:

\[
\text{Set} = \text{Set}^{\circ}
\]

II.0.10 Definition [ ] The standard semi-cube \( n \) is defined as the Yoneda embedding of \( n \in \text{Obj} \):

\[
\text{n Set} \quad n (p) = \text{Hom} (p,n) \\
n (f) = \lambda g \cdot g \circ f
\]

Standard \( n \)-semi-cubes have a geometric interpretation, which we illustrate for dimensions 0, 1, and 2.

II.0.11 Example [ ] \( 0 \) is \( \text{Hom} \ 0,0 \), or the singleton set of the empty sequence:

II.0.12 Example [ ] \( 1 \) consists of two points, given by \( \text{Hom} \ 0,1 \), and a line, given by \( \text{Hom} \ 1,1 \).

\[
L \rightarrow R
\]

II.0.13 Example [ ] \( 2 \) consists of four points, given by \( \text{Hom} \ 0,2 \), four lines connecting the four points, given by \( \text{Hom} \ 1,2 \), and a filler, given by \( \text{Hom} \ 2,2 \):

\[
\begin{array}{c}
\text{LR} \rightarrow \text{RR} \\
L \ast \quad \ast \ast \quad R \ast \\
\text{LL} \quad \ast \ast \quad \ast \ast
\end{array}
\]
More generally, the standard $n+1$-semi-cube can be obtained by taking two copies of the standard $n$-semi-cube serving as bottom and top face and connecting them on their border by a cylinder obtained as a third copy stretched in the new dimension. The bottom and top faces are obtained from the standard $n$-semi-cube by postfixing with respectively $L$ and $R$ while the cylinder is obtained by postfixing with $\ast$. Note that the components can be oriented by letting each $n$-dimensional component go from the $n-1$-dimensional component obtained by replacing the leftmost $\ast$ with $L$, to the one obtained by replacing the leftmost $\ast$ with $R$.

$\nu$-sets

Let us call $\nu$-sets, the generalization of augmented semi-simplicial sets and semi-cubical sets obtained by building on an arbitrary alphabet $\nu$, so that the following holds:

<table>
<thead>
<tr>
<th>Cardinal of $\nu$</th>
<th>$1$</th>
<th>$2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretation</td>
<td>augmented semi-simplicial types</td>
<td>Semi-cubical types</td>
</tr>
</tbody>
</table>

To obtain this, we extend $+$ and $\cdot$ in a straightforward manner into a category which we call $\nu$-semi-shape category is shown below. Note that, if $g \circ f$ is well-defined, then the length of $f$ is less than that of $g$. It can be shown that composition is associative and that id is neutral.

\[
\begin{align*}
\text{Obj} &= \{ l \in \nu \uplus \{\ast\}^n \mid \text{number of } \ast \text{ in } l = p \} \\
\text{Hom}_{p,n} &= \begin{cases}
  f & \text{if } g = \\
  a \cdot g' \circ f' & \text{if } g = a g', \text{where } a \in \nu \\
  a \cdot g' \circ f' & \text{if } g = \ast g', f = a f', \text{where } a \in \nu \text{ or } a = \ast
\end{cases}
\]

\[
\text{id} = \ast \ldots \ast n \text{ times for } \text{id} \in \text{Hom}_{n,n}
\]

$\nu$-set is thus a contravariant functor $\phi$ from the $\nu$-semi-shape category to $\text{Set}$ and we call $n-\nu$-semi-shape an element of $\phi$ $n$.

III Type theory

Martin-Löf’s Type theory ([ML75, Mar84]) is a logical formalism based on the notion of a type rather than that of a set. It can be seen as a foundation of mathematics alternative to set theory and is the core of several tools for the formalization of mathematics such as gda ([The23a]), Coq ([The23b]) and Lean ([dMK +15]). In type theory, propositions are types and proofs are programs. Type theory includes definitional equality, by which all propositions and proofs are quotiented.

Type theory is a flexible formalism supporting different models. Some models are based on topological spaces, where equality is interpreted as path, and substitutivity of equality as transport [KL21]. These models support the univalence principle stating that equality of types mimics equivalence of types, leading to the development of Homotopy Type Theory [The13].

Types are organized in a hierarchy of universes written $\text{Type}_m$ for $m$ a natural number. The main types in type theory are the type of dependent pairs, written $\Sigma a \ A \ B \ a$, the type of dependent functions, written $\Pi a \ A \ B \ a$, for $A$ a type and $B$ a $a$ a type dependent on the inhabitant $a$ of $A$, and the type of propositional equalities, written $t = u$. s a notation, the type of dependent pairs when $B$ is not dependent on $A$ is shortened into $A \times B$ and the type of dependent functions when $B$ is not dependent on $A$ is written $A \rightarrow B$. We assume our type theory to also include a distinguished singleton type, written unit, and with inhabitant $\ast$, the type of boolean values, and the type of natural numbers. We also assume that our type theory includes the coinductive type of infinite dependent tuples. We write $\text{hd}$ and $\text{tl}$ the projections of dependent pairs, and $\text{refl}$ for reflexivity. Logical propositions being types themselves, we use $\Pi$ to represent universal quantification and $\Sigma$ to represent existential quantification.

type-theoretic notion of sets can be recovered in each universe as $\text{HSet}_m$, denoting the subtype of $\text{Type}_m$ for which paths are degenerated, using Uniqueness of Identity Proofs (UIP). Technically, this is expressed as a
structure equipping a domain $\text{Dom}$ with the property UIP:

\[
\text{Dom} \quad \text{Type}_m \\
\text{UIP} \quad 1 \forall y \quad \text{Dom}.1 \forall p \quad x = y, p = q
\]

In $\text{HSet}_m$, the following properties hold:
1. UIP holds on the unit type, bool type, as well as all types of finite cardinal $\nu$.
2. UIP propagates to $\Sigma$-types.
3. UIP propagates to $\Pi$-types, with some additional functional extensionality axioms.
   
By notation, $\text{Type}$ and $\text{HSet}$ mean $\text{Type}_m$ and $\text{HSet}_m$ at some unspecified universe level $m$.

We are also interested in extensional type theory, a type theory with the following reflection rule, where $=$ is propositional equality in some type and $\equiv$ is definitional equality [Mar84]:

\[
\vdash p \quad t = u \\
\vdash t \equiv u
\]

Note that the reflection rule implies UIP so that $\text{HSet}$ and $\text{Type}$ are equivalent in extensional type theory. The reflection rules also imply functional extensionality. Extensional type theory is logically equivalent to intensional type theory extended with UIP and functional extensionality [Hof95].

**IV Relation to P R Metricity**

Recall from the introduction, the form taken by the indexed presentation of a semi-cubical set:

\[
X_0 \\
X_1 \\
X_2 \quad \Pi abcd. \quad X_1 \times X_1 \times X_1 \times X_1 \times X_1 \\
\ldots
\]

Here, the process of construction of the type of $X_1$ from that of $X_0$, and of the type of $X_2$ from that of $X_1$, is similar to iteratively applying a binary parametricity translation. The binary parametricity which we consider interprets a closed type $A$ by a family $A_T$ over $A \times A$, and this can be seen as a graph whose vertices are in $A$. Each type constructor is associated with the construction of a graph. To start with, the type of $\text{HSet}$ is interpreted as the family of type of families $\text{HSet}_*$, which takes $A_L$ and $A_R$ in $\text{HSet}$ and returns the type $A_L \times A_R \to \text{HSet}$ of families over $A_L$ and $A_R$. Also, for $A$ interpreted by $A_T$ and $B$ interpreted by $B_T$, a dependent function type $\Pi a \quad A(a) \times B(a)$ is interpreted as the graph $\Pi a \quad A(a) \times B(a)$, that takes two functions $f_L$ and $f_R$ of type $\Pi a \quad A(a) \times B(a)$, and expresses that these functions map related arguments in $A$ to related arguments in $B$:

\[
\Pi a \quad A(a) \times B(a) \rightarrow \Pi a \quad A(a) \times B(a)
\]

Similarly, a product type $A \times B$ is interpreted as the graph $\Pi a \quad A(a) \times B(a)$, that relates two tuples $(a_L, b_L)$ and $(a_R, b_R)$ in $A \times B$ as follows:

\[
A \times B \rightarrow \Pi a \quad A(a) \times B(a)
\]

In particular, for $X \quad \text{HSet}$, applying our parametricity translation is about associating to $X$ an inhabitant $X_*$ of $\text{HSet}_* (X, X)$ i.e. of $X \times X \rightarrow \text{HSet}$. In turn, applying the translation again to $X_*$, $X_\ast \times X_\ast \rightarrow \text{HSet}$ is about associating to $X_*$ an inhabitant $X_*\ast$ of $X_\ast \times X_\ast \rightarrow \text{HSet}$. This tells us how the sequence $X_0, X_1, X_2$ can be seen as a sequence of inhabitants of the iteration of the composition of binary parametricity with the diagonal on types and type families, applied to an initial $X \quad \text{HSet}$:

\[
X_0 \triangleq X \\
X_1 \triangleq X_\ast \quad \text{HSet}_* (X, X) \\
X_2 \triangleq X_\ast \ast \quad \text{HSet}_* (X, X)_\ast \quad X_\ast \ast, X_\ast
\]

This tells us how the informal type given to $X_2$ in the previous section could be rephrased so that it comes as the instance of a general recipe characterizing the type of all $X_i$. 
Notice, however, that the recipe obtained so far, \( \langle X_{n+1}, S_n \rangle, X_n, X_n \rangle \) for \( X_n \) \( S_n \), applies parametricity on the syntax of the type of \( X_n \). It does not directly yield a characterization of \( S_n \) as a function from \( n \). Reformulating the recipe as an explicit recursive construction, without requiring an interpretation of the syntax of types, is the main outcome of this work, together with the mechanization and the uniform treatment of augmented semi-simplicial and semi-cubical sets by means of the generalization to \( \nu \)-sets.

V Our construction

In this section, we describe our parametricity-based construction of \( \nu \)-sets in indexed form at two levels of formality.

Sections V.1 and V.3 describe the construction at an informal level of discourse:

1. In V.1, we present it in informal extensional type theory where equational reasoning is left implicit, and we give an intuition for the construction in V.2.
2. While reasoning in extensional type theory is similar to reasoning in set theory regarding how equality is handled, extensional type theory has two limitations. The first limitation is that it enforces the principle of Uniqueness of Identity Proofs and this is inconsistent with the Univalence principle, thus making it inexpressible in Homotopy Type Theory. The second limitation is that we want the construction to be formalizable in the Coq proof assistant whose underlying type theory is intensional. Section V.3 thus rephrases the construction in (informal) intensional type theory.

Sections V.4, V.5, and V.6 describe additional issues to be addressed in order to get a fully formal construction:

1. The well-foundedness of the induction requires a special termination evidence which will be discussed in section V.4.
2. The construction is indexed over integers and holds under some constraints on the range of these integers. There is a standard formalization dilemma in this kind of situation: either the constraints on the range are embedded in the construction so that the construction makes sense only on the corresponding range, or the construction is made first on a more general domain than needed but restricted to a smaller domain at the time of use. We adopted the former approach, requiring the construction to be dependent on proofs of inequalities on natural numbers. We discuss how we deal with such dependencies in section V.5.
3. number of standard groupoid properties of equality as well as type isomorphisms have been left implicit in the informal definition. This is discussed in section V.6.

V.1 The construction in informal type theory

\( \nu \)-set in indexed form is a sequence of families of \( \text{HSet} \), that is \( \text{HSet}_m \) for some universe level \( m \). We call such sequence a \( \nu \)-set at level \( m \), whose type thus lives in \( \text{HSet}_{m+1} \).

Table 1 describes the type of a \( \nu \)-set at level \( m \) in indexed form, as the type of a coinductively-defined infinite sequence of type families representing the limit of \( n \)-truncated \( \nu \)-sets: \( \nu \text{Set}^{< n}_m \) denotes an infinite sequence \( X_{n+1}, X_{n+2}, \ldots \) dependent on a \( n \)-truncated \( \nu \)-set, \( \nu \text{Set}^{< n}_m \), so that, when \( n = 0 \), it denotes a full \( \nu \)-set, written \( \nu \text{Set}_m \). This is made possible because the 0-truncated \( \nu \)-set, \( \nu \text{Set}^{0}_m \), is degenerated: it is an empty family, and there is thus only one 0-truncated \( \nu \)-set, namely the canonical inhabitant \( \ast \) of unit.

The definition of the type of a \( n \)-truncated \( \nu \)-set is in turn described in table 2. In the infinite sequence of type families representing a \( \nu \)-set, the \( n \)-th component is a type dependent over a fullfr \text{me}. It is recursively defined in table 3, using the auxiliary definitions of \text{fr me}, \text{l yer} and \text{p inting}. fullfr \text{me} describes a boundary of a standard form (simplex, cube), which we decompose into \text{l yer}, and a \text{p inting} corresponds to a filled frame. Notice that the type \text{l yer} relies on an operator of frame restriction \text{restr} \text{fr me} which is defined in table 4, and this restriction operator is in turn defined using auxiliary definitions \text{restr} \text{fr me} and \text{restr} \text{p inting}.

Notably, the definition of \text{restr} \text{fr me} relies on an equality expressing the commutation of the composition of two \text{restr} \text{fr me}. The proof of this commutation is worth being made explicit, which we do in table 5 using proof-term notations. The proof requires an induction on the dimension and on the structure of \text{fr me}, \text{l yer}, and \text{p inting}. This is what \text{coh} \text{fr me} does using auxiliary proofs \text{coh} \text{l yer} and \text{coh} \text{p inting}. Even though it looks independent of the definitions from the other tables, \text{coh} \text{fr me} has to be proved mutually with the definitions of \text{fr me}, \text{l yer}, \text{p inting}, and their corresponding restrictions. More precisely, for a fixed \( n \), the block of \text{fr me}, \text{restr} \text{fr me}, and \text{coh} \text{fr me} has to be defined in one go by induction on \( p \). Also, each of \text{p inting}, \text{restr} \text{p inting}, and \text{coh} \text{p inting} is built by induction from \( p \) to \( n \). The \text{p inting} block at \( n \) relies on the \text{fr me} block at \( n \), but the converse dependency is only on lower \( n \), so this is well-founded. Note that \text{l yer}, \text{restr} \text{l yer} and \text{coh} \text{l yer} are just abbreviations. The exact way this mutual recursion is eventually formalized is explained in section V.4.

Note that for a fixed constant \( n \), relying on the evaluation rules of type theory, the coherence conditions degenerate to a reflexivity proof, so that the construction builds an effective sequence of types not mentioning coherences anymore.
Table 1: Main definition

<table>
<thead>
<tr>
<th>$\nu\text{Set}_m$</th>
<th>$\triangle$</th>
<th>$\nu\text{Set}_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu\text{Set}^{&gt;\ast}_m$</td>
<td>$\nu\text{Set}^{&gt;\ast}_m$</td>
<td></td>
</tr>
<tr>
<td>$\nu\text{Set}^{&gt;\ast}_m$</td>
<td>$\nu\text{Set}^{&gt;\ast}_m$</td>
<td></td>
</tr>
</tbody>
</table>

$\vdash \exists \nu\text{Set}_m \exists \nu\text{Set}^{>\ast}_m \exists \nu\text{Set}^{>\ast}_m \in \text{CHSet} \ni \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

$\Sigma H \nu\text{Set}_m \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

Table 2: Truncated $\nu$-sets, the core

<table>
<thead>
<tr>
<th>fullfr $\text{me}_n$</th>
<th>$\nu\text{Set}_m$</th>
<th>$H\text{Set}_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullfr $\text{me}_n$</td>
<td>$\nu\text{Set}_m$</td>
<td>$H\text{Set}_m$</td>
</tr>
</tbody>
</table>

$\vdash \exists \nu\text{Set}_m \exists \nu\text{Set}^{>\ast}_m \exists \nu\text{Set}^{>\ast}_m \in \text{CHSet} \ni \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

$\Sigma D \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

Table 3: fr me, l yer, and p inting

<table>
<thead>
<tr>
<th>restr$^{n,p,q}_{p \text{ inting}, q}$</th>
<th>$\nu\text{Set}_m$</th>
<th>$H\text{Set}_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>restr$^{n,p,q}_{p \text{ inting}, q}$</td>
<td>$\nu\text{Set}_m$</td>
<td>$H\text{Set}_m$</td>
</tr>
</tbody>
</table>

$\vdash \exists \nu\text{Set}_m \exists \nu\text{Set}^{>\ast}_m \exists \nu\text{Set}^{>\ast}_m \in \text{CHSet} \ni \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

$\Sigma D \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

Table 4: $q$-th projection of restr, or faces

<table>
<thead>
<tr>
<th>co$^{n,p,q}_{q \text{ inting}, \omega, q}$</th>
<th>$\nu\text{Set}_m$</th>
<th>$H\text{Set}_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>co$^{n,p,q}_{q \text{ inting}, \omega, q}$</td>
<td>$\nu\text{Set}_m$</td>
<td>$H\text{Set}_m$</td>
</tr>
</tbody>
</table>

$\vdash \exists \nu\text{Set}_m \exists \nu\text{Set}^{>\ast}_m \exists \nu\text{Set}^{>\ast}_m \in \text{CHSet} \ni \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

$\Sigma D \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m \nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$ $\nu\text{Set}^{>\ast}_m$

Table 5: Commutation of $q$-th projection and $r$-th projection, or coherence conditions
V.2 Intuition for our formal construction

There is a \textit{fullfr me} for each dimension \( n \), written \( \text{fullfr me}^n \), and every \( X_n \) is uniformly assigned a type of the form \( \text{fullfr me}^n \to \text{HSet} \). Here, \( \text{fullfr me}^n \) is a “telescope” collecting all arguments of the type of \( X_i \) in section IV as a nesting of \( \Sigma \)-types.

To illustrate how to recursively build \( \text{fullfr me}^n \), let us begin by setting \( \text{fullfr me}^0 \equiv \text{unit} \), so that the type \( \text{HSet} \) given to \( X_0 \) in section IV can be equivalently formulated as \( \text{unit} \to \text{HSet} \). Then, more generally, let each \( \text{fullfr me}^n \) consist of \( n \) layers, written \( l \\text{yer}^{n,p} \) with \( p < n \), that we stack in order, starting from unit, and writing \( \text{fr me}^{n,p} \) for the \( p \) first layers of a \( \text{fullfr me}^n \), so that \( \text{fullfr me}^n \) is \( \text{fr me}^{n,n} \). For instance, \( X_1 \) is made of one layer, so that it can be written as a \( \Sigma \)-type of an inhabitant of unit and \( l \\text{yer}^{1,0} \). Then, \( X_2 \) is similarly made of two layers.

\[
\begin{align*}
X_0 \quad \text{unit} & \to \text{HSet} \\
X & \quad \Sigma^+ \quad \text{unit.} \\
X_2 \quad \Sigma^\alpha & \quad \Sigma^+ \quad \text{unit.} \\
\end{align*}
\]

Let us now illustrate the construction of \( \text{fullfr me}^3 \), necessary to build the type of \( X_3 \).

The figure on the left is \( \text{fr me}^{3,1} \), in the middle is \( \text{fr me}^{3,2} \), and on the right is \( \text{fr me}^{3,3} \), which is full. Further, \( \text{fr me}^{3,1} \) is made of one layer, \( l \\text{yer}^{3,0} \), shown in blue, \( \text{fr me}^{3,2} \) is made of one additional layer, \( l \\text{yer}^{3,1} \), shown in red, \( \text{fr me}^{3,3} \) is made of one more layer, \( l \\text{yer}^{3,2} \), shown in green.

We illustrated here the cubical case, that is \( \nu = 2 \), but, in general, a \( l \\text{yer}^{n,p} \) is a product of \( \nu \ \text{p inting}^{n,1,p} \). \( \text{p inting}^{n,0} \) is a \( n \)-dimensional object corresponding to a filled \( \text{fullfr me}^n \). More generally, a \( \text{p inting}^{n,p} \) is an \( n \)-dimensional object which has the form of a \( \text{p inting}^{n,0} \), thus of \( n \)-dimensional form, but shifted and living in dimensions \( p \) to \( n \). Such \( \text{p inting}^{n,p} \) fills a space framed by a partial \( \text{fr me}^{n,p} \) so that, together, they
form a filled \textit{fullfr me} \textsuperscript{n}. For instance, in the picture, each of the two \textit{p inting} \textsuperscript{2,0} of \textit{l yer} \textsuperscript{2,0} is a filled blue square, each of the two \textit{p inting} \textsuperscript{2,1} of \textit{l yer} \textsuperscript{3,1} is a line, shown in red, stretched into a partial square filling the partial frames made of respectively, the left and right border of the blue square, and each of the two \textit{p inting} \textsuperscript{2,2} of \textit{l yer} \textsuperscript{3,2} is the point shown in green, stretched into a partial square filling the full frames made respectively of the upper and lower borders of the blue and red squares. \textit{p inting} \textsuperscript{n,p} complements a \textit{fr me} \textsuperscript{n,p} by adding layers needed to form a \textit{fullfr me} \textsuperscript{n} and by filling the resulting \textit{fullfr me} \textsuperscript{n} with an inhabitant of \textit{X}_n. Layers are added from dimension \textit{n} to dimension \textit{p}, opposite to the order from \textit{p} to \textit{fr me} \textsuperscript{n,p} are built, as shown below.

\[
\begin{align*}
\textit{fr me}^{n,p} & \triangleq \Sigma_{a_n} \ldots \Sigma_+ \text{ unit}. \textit{yer}^{n,0}(0) \ldots \textit{yer}^{n,p} \ 1 \\
\textit{p inting}^{n,p} & \triangleq \Sigma_{p \ l \ yer}^{n,p} \ldots \Sigma_{l_n}^{n \ l \ yer, n} \ (X_n) \ldots
\end{align*}
\]

So far, we have not paid attention to the fact that we have a dependent type, shown as \textit{\Sigma}. To be more precise, note that \textit{fullfr me} \textsuperscript{n} depends on all \textit{X}_i up to \textit{n} = 1. So, we need to package up \textit{X}_i for \textit{i} < \textit{n}, into a nesting of \textit{\Sigma}-types, constituting the type of a \textit{n}-truncated \textit{\nu}-set, which we wrote \textit{\nuSet} \textsuperscript{<n}. This allows us to give the type \textit{\nuSet} \textsuperscript{<n} \rightarrow \textit{HSet} to \textit{fullfr me} \textsuperscript{n}. Then, for \textit{D} \ \textit{\nuSet} \textsuperscript{<n}, representing an initial prefix of \textit{X}_0, \textit{X}_1, \ldots \textit{X}_n 1, the indexed set \textit{X}_n has type \textit{fullfr me} \textsuperscript{n} \textit{D}) \rightarrow \textit{HSet}. Thus, \textit{fr me} \textsuperscript{n,p}, \textit{l yer} \textsuperscript{n,p} and \textit{p inting} \textsuperscript{n,p} also depend on \textit{D}. We can then reformulate the previous equation with its dependency on \textit{D}. In particular, \textit{X}_n is just the last component of \textit{D}, that is \textit{D}.tl.

\[
\begin{align*}
\textit{fr me}^{n,p} D) \triangleq \Sigma_{a_n} \ldots \Sigma_+ \text{ unit}. \textit{yer}^{n,0}(D) \ldots \textit{yer}^{n,p} \ 1 D) \\
\textit{p inting}^{n,p} D) d) \triangleq \Sigma_{p \ l \ yer}^{n,p} \ldots \Sigma_{l_n}^{n \ l \ yer, n} \ D) (D).tl d) \ldots
\end{align*}
\]

\textit{n} extra refinement arises from the fact that each new layer of a frame has to be glued onto the border of the partial frame built so far. So, each \textit{l yer} \textsuperscript{n,p} has to depend on \textit{fr me} \textsuperscript{n,p}. We also need a way to characterize the \textit{\nu} borders of each \textit{p inting} \textsuperscript{1,p} that composes a \textit{l yer} \textsuperscript{n,p}, and this is where the restriction \textit{rest} \textsuperscript{fr me, \ l \ yer} \textsuperscript{n,p} arrives, for each < \textit{\nu}. For instance, in the picture, the left and right and \textit{p inting} \textsuperscript{2,1}, shown in red, are laid on respectively the left and right borders of the blue boxes, and hence need to depend on \textit{fr me} \textsuperscript{2,1}. The left and right borders of the two blue boxes are then extracted as \textit{rest} \textsuperscript{2,1} \textit{fr me, L} \textit{D}) \textit{d) and \textit{rest} \textsuperscript{2,1} \textit{fr me, R} \textit{D}) \textit{d). We can then refine again the previous equation by showing the dependencies on \textit{D}, as shown below.

\[
\begin{align*}
\textit{fr me}^{n,p} D) \triangleq \Sigma_{a_n} \ldots \Sigma_+ \text{ unit}. \textit{yer}^{n,0}(D) \ldots \textit{yer}^{n,p} \ 1 D) \\
\textit{p inting}^{n,p} D) d) \triangleq \Sigma_{p \ l \ yer}^{n,p} \ldots \Sigma_{l_n}^{n \ l \ yer, n} D) d) \ldots \Sigma d \ldots \Sigma_+ \text{ unit}. \textit{yer}^{n,0} (D) \ldots \textit{yer}^{n,p} (D) d) \\
\text{where d, l_p, \ldots, l_q abbreviates d, l_p, \ldots, l_q)
\end{align*}
\]

When \textit{\nu} = 2, using \textit{L} and \textit{R} to represent the sides, the formation of layers from paintings amounts to:

\[
\begin{align*}
\textit{l yer}^{n,p} D) \textit{d) \triangleq \textit{p inting}^{n,1,p} D.\textit{hd}) \textit{rest}^{\textit{fr me, L} \textit{p}} \textit{d}) \times \textit{p inting}^{n,1,p} D.\textit{hd}) \textit{rest}^{\textit{fr me, R} \textit{p}} \textit{d})
\end{align*}
\]

The operation \textit{rest} \textsuperscript{\textit{fr me, q}} restricts the \textit{p} first layers of a frame, and the construction is by recursion on the structure of a frame \textit{d}. This necessitates the definitions \textit{rest} \textsuperscript{\textit{yer}, q} \textit{d}) \textit{d}) and \textit{rest} \textsuperscript{\textit{p inting}, q} \textit{d}) \textit{c}), for \textit{d} a \textit{l yer}, and \textit{c} a \textit{p inting}. The key case is \textit{rest} \textsuperscript{\textit{p inting}, d}) \textit{d}) \textit{c}), where \textit{c}, a \textit{p inting}, necessarily has the form of \textit{c_L, c_R, \ldots}). Here, \textit{rest} \textsuperscript{\textit{p inting}, \textit{L} \textit{p}} \textit{d}) picks out \textit{c_L}, a \textit{p inting} \textsuperscript{1,p}, \textit{rest} \textsuperscript{\textit{p inting}, \textit{R} \textit{p}} \textit{d}) picks out the \textit{c_R}, also a \textit{p inting} \textsuperscript{1,p}, and the last component, shown as \ldots, a \textit{p inting} \textsuperscript{n,p+1}, is discarded. There is one last difficulty, which we illustrate by writing down expected and actual types.

Given \textit{c} of type

\[
\begin{align*}
\textit{p inting}^{n,1,p} D.\textit{hd}) \textit{rest}^{\textit{fr me, \omega q}} \textit{d})
\end{align*}
\]

\textit{rest} \textsuperscript{\textit{yer}, d}) \textit{c_L, c_R}) produces a layer in which the \textit{\omega}-component has the type

\[
\begin{align*}
\textit{p inting}^{n,2,p} D.\textit{hd,hd}) \textit{rest}^{\textit{fr me, q}} \textit{rest}^{\textit{fr me, \omega q}} d)))
\end{align*}
\]

while we expect a component of type

\[
\begin{align*}
\textit{p inting}^{n,2,p} D.\textit{hd,hd}) \textit{rest}^{\textit{fr me, q}} \textit{rest}^{\textit{fr me, q+1 d))}
\end{align*}
\]

Hence, we need a coherence condition to commute the restrictions. Coherence conditions similar to this necessitate what are shown as \textit{coh} \textsuperscript{fr me}, \textit{coh} \textsuperscript{yer} and \textit{coh} \textsuperscript{p inting} in table 5. These are by induction on the structure of \textit{fr me}, \textit{yer} and \textit{p inting}. Note that, for the construction in intensional type theory, we further need a 2-dimensional coherence condition, \textit{coh} \textsuperscript{2fr me}, for \textit{coh} \textsuperscript{yer}, which is explained in the next section.
V.3 From extensional to intensional type theory

In this section, we intend to get rid of the reflection rule and make explicit the equational reasoning step needed to rephrase the construction in intensional type theory. For readability purposes, we make only explicit in this section the key coherence conditions of the construction. Other cases of equality reasoning would have to be made explicit to fully obtain a construction in intensional type theory, but these steps are standard enough to be omitted at this stage. See section V.6 for the details.

The need for transport along a proof of commutation of \(\text{restr}_{\text{fr me}}\) in the definition of \(\text{restr}_{\text{fr yer}}\) is made explicit in table 4’, where the arrow over \(\text{coh}_{\text{fr me}}\) indicates the direction of rewrite.

<table>
<thead>
<tr>
<th>Table 4’: q-th projection of (\text{restr}), or faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{restr}_{\text{fr me}}{n \leq n} \quad {D \ \text{Set}^{\leq n}} \quad {D \ \text{fr me}^{n-p} D} \quad D \ d \ l \ \triangleq \ \Delta )</td>
</tr>
<tr>
<td>(\text{restr}_{\text{fr me}}{n} \quad {D \ \text{Set}^{\leq n}} \quad {D \ \text{fr me}^{n-p} D} \quad D \ d \ l \ \triangleq \ \Delta )</td>
</tr>
<tr>
<td>(\text{restr}_{\text{fr me}}{n} \quad {D \ \text{Set}^{\leq n}} \quad {D \ \text{fr me}^{n-p} D} \quad D \ d \ l \ \triangleq \ \Delta )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5’: Commutation of q-th projection and r-th projection, or coherence conditions</th>
</tr>
</thead>
</table>
| The proof of \(\text{coh}_{\text{fr me}}\) itself requires making explicit several rewrites which were invisible in extensional type theory. The commutation of \(\text{restr}_{\text{fr me}}\) lives in a type referring to \(\text{coh}_{\text{fr me}}\), so we need a transport along the commutation of \(\text{restr}_{\text{fr me}}\) in the statement of \(\text{coh}_{\text{fr me}}\). The proof of \(\text{coh}_{\text{fr me}}\) itself is the most involved proof of the construction, as it requires a higher-dimensional coherence condition, \(\text{coh}_{\text{fr me}}\), whose exact formulation is as follows.

\[
\begin{align*}
\text{coh}_{\text{fr me}}^{n,p} &\mid n \leq n \quad \{D \ \text{Set}^{\leq n}\} \quad \{D \ \text{fr me}^{n-p} D\} \quad D \ d \ l \ \Delta \ \Delta \\
\text{p restr}_{\text{fr me}}^{n,p} &\mid n \leq n \quad \{D \ \text{fr me}^{n-p} D\} \quad D \ d \ l \ \Delta \\
\text{restr}_{\text{fr me}}^{n,p} &\mid n \leq n \quad \{D \ \text{fr me}^{n-p} D\} \quad D \ d \ l \ \Delta \\
\end{align*}
\]

where \(\text{p}\) applies a function on two sides of an equality, and \(\bullet\) is transitivity of equality. This property of equality proofs holds in \(\text{HSet}\), and since our construction is done in \(\text{HSet}\), the term is trivially discharged.
Notice that each \( \text{restr}_{\text{fr me}} \) in the type of \( \text{coh}_{\text{fr me}} \) is hiding a \( \text{coh}_{\text{fr me}} \) rewrite: this makes a sum total of three \( \text{coh}_{\text{fr me}} \) rewrites on the left-hand side, and two \( \text{coh}_{\text{fr me}} \) rewrites on the right-hand side. In the proof term of \( \text{coh}_{\text{fr me}} \), \( \text{coh}_{\text{fr me}} \) has one \( \text{coh}_{\text{fr me}} \) rewrite on its left-hand side. This, combined with the two terms of the form \( \text{p coh}_{\text{fr me}} \), matches our expectation of three \( \text{coh}_{\text{fr me}} \) on the left-hand side, and two \( \text{coh}_{\text{fr me}} \) on the right-hand side. Then, \( \text{coh}_{\text{fr me}} \) can be seen as expressing the commutation of these \( \text{coh}_{\text{fr me}} \) terms.

Finally, let us explain \( \text{coh}_{\text{p inting}} \). The base case \( p = r \) is the key case of the commutation of \( \text{restr}_{\text{fr me}} \), when one of the \( \text{restr}_{\text{p inting}} \) collapses, and the remaining equation holds trivially. The case of \( p < r \) follows the structure of \( \text{restr}_{\text{p inting}} \) by induction.

If we were not working in \( \text{HSet} \), but in \( \text{HGpd} \) we would need to prove one more higher-dimensional coherence, and if we were working in \( \text{Type} \), we would need to prove arbitrarily many higher-dimensional coherences. Here, \( \text{HGpd} \) is the subset of types \( A \) such that for all \( x \) and \( y \) in \( A \), \( x = y \) is in \( \text{HSet} \). See [Her15, CK16, Kra21] for a discussion on the need for recursive higher-dimensional coherence conditions in formulating higher-dimensional structures in type theory.

V.4 Well-foundedness of the construction

Since the construction shown in the previous sections is by induction on \( n \), and dependencies are on lower \( n \) and \( p < n \), one would imagine formalizing this using well-founded induction in dependent type theory. We initially tried this approach, and had terms dependent on the proofs of the case distinction that \( n' \leq n \) implies \( n' < n \) or \( n' = n \), but these proofs did not come with enough definitional properties to be usable in practice. Hence, we chose a different route: in practice, since \( \text{restr}_{\text{fr me}} \) depends on \( \text{fr me}^n \) and \( \text{fr me}^{n-1} \), while \( \text{coh}_{\text{fr me}} \) depends on \( \text{fr me}^n \), \( \text{fr me}^{n-1} \), and \( \text{fr me}^{n-2} \), we only need to keep track of three consecutive dimensions. Hence, what we build by induction at level \( n \), is a structure made not only of the definitions shown in the tables 3, 4', and 5', but also of \( \text{fr me}, \text{p inting}, \text{leR}, \text{leR refl} \) at levels \( n = 1 \) and \( n = 2 \), as well as \( \text{restr}_{\text{fr me}}, \text{restr}_{\text{p inting}} \) at level \( n = 1 \), together with helper equations.

V.5 Dependencies in inequality proofs

The entire construction relies on inequalities over natural numbers, and we use two different definitions of \( \leq \) addressing different concerns in our formalization. In order to build our first variant, we use an intermediate “recursive definition” phrased as:

\[
\text{Fixpoint } \text{leR} \ (n \ m : n t) : \text{SProp} := \\
\begin{array}{ll}
\text{m tch n, m with} \\
| \text{U, _ } \Rightarrow \text{STrue} \\
| \text{S n, O } \Rightarrow \text{SF lse} \\
| \text{S n, S m } \Rightarrow \text{leR n m} \\
\end{array}
\]

Here, \( \text{SProp} \) is a distributionally proof-irrelevant impredicative universe morally\(^4\) living at the bottom of the universe hierarchy [GCST19]. By placing the definition in \( \text{SProp} \), we have definitional equality of inequality proofs. However, for the purpose of unification, this definition does not go far enough. Consider the unification problems:

\[
\text{leR tr ns } ?p \text{ leR refl } = ?p \\
\text{leR tr ns leR refl } ?p = ?p
\]

where \( \text{leR tr ns} \) is transitivity, \( \text{leR refl} \) is reflexivity, and \( ?p \) is an existential variable. These two problems definitionally hold in \( \text{SProp} \), but equating them does not solve the existential variable. For unification to be useful in solving existential variables, we present our first variant of \( \leq \), which we dub as the “Yoneda variant”:

**Definition** \( \text{leY} \ n m := \\
\begin{array}{ll}
\text{for } l p, \text{leR p n } \Rightarrow \text{leR p m.} \\
\end{array}
\]

This definition is an improvement over \( \text{leR} \) since reflexivity is now definitionally the neutral element of transitivity, and associativity of transitivity also holds definitionally. Although it significantly eases our proof, there are some instances where unification is unable to solve the existential variables, and we have to provide them explicitly.

The second variant of \( \leq \), the “inductive variant”, is phrased as:

\(^4\)In Coq ([The23b]), it is however a stand-alone universe unrelated to the universe hierarchy.
**Inductive** leI : n t -> n t -> Type :=
| leI_refl n : n < n
| leI_down (n p) : p.+1 <- n -> p <- n
where "n <- m" := (leI n m) : n t_scope.

Compared to leY, leI has no proof-irrelevance properties. This definition is specially crafted for **p inting**, where we have to reason inductively from \( p \leq n \) to \( n \). In our usage, we have lemmas leY_of_leI and leI_of_leY in order to equip leY with the induction scheme of leI. The resulting induction scheme has computational rules holding propositionally.

### V.6 Groupoid properties of equality and basic type isomorphisms

The construction relies on groupoid properties of equality which are left implicit in table 5’. The use of the equivalence between \( u = v \) and \( \Sigma p \ u.hd = v.hd \). \( u \:: \:: v \:: \) for \( u \) and \( v \) in a \( \Sigma \)-type is left implicit in the same table. Iso implicit is the use of the equivalence between \( f = g \) and \( \Pi a \ A. f a = g a \) for \( f \) and \( g \) in \( \Pi a \ A.B \), where it should be recalled that the right-to-left map, or functional extensionality, holds by default in extensional type theory. These have to be made explicit in the formalization.

A final remark, note that as a consequence of \( \eta \)-conversion for finite enumerated types, the requirement of functional extensionality disappears when \( \nu \) is finite. However, this is a conversion which Coq does not implement, and the alternative would be to replace \( \Pi a \ \nu. B \) by a “flat” iterated product \( B \prod 1 \times B \prod 2 \times \ldots \times B \nu \).

## VI Future work

The construction could be extended with degeneracies as well as with permutations [GM03]. Dependent \( \nu \)-sets could also be defined, opening the way to construct \( \Pi \)-types and \( \Sigma \)-types of \( \nu \)-sets. \( \nu \)-set of \( \nu \)-sets representing a universe could also be defined as sketched in a talk at the HoTT-UF workshop for the bridge case ([Her20]).

More generally, we believe these lines of work would eventually provide alternative models to parametric type theories [NVD17, CH20] where equality of types, now a family rather than the total space of a fibration, is not only definitionally isomorphic to bridges [BCM15], but definitionally the same as bridges.

By equipping the universe construction with a structure of equivalences, as suggested along the lines of [K15], we also suspect the construction to be able to serve as a basis for syntactic models of various versions of cubical type theory [BCH13, CHHM18, BC+21], saving the detour via the fibered approach inherent to usual presheaf models. In particular, we conjecture being able to justify univalence holding definitionally. Our approach would also firmly ground cubical type theory in iterated parametricity.

Another direction for future work is to give a general definition of presheaf in indexed form over a direct category, and to show the equivalence with the standard definition of presheaf.

## References


REFERENCES

[15/15]


