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

Hugo Herbelin and Ramkumar Ramachandra

Université Paris Cité, Inria, CNRS, IRIF, Paris Hugo.Herbelin@inria.fr
Université Paris Cité (2020-2022), Unaffiliated r@artagnon.com

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 popularized an alternative definition, where the set of n-simplices or n-cubes are instead regrouped into the families of the fibers over their faces, leading to a characterization we call *indexed*. Moreover, it is known that semi-simplicial and semi-cubical sets are related to iterated Reynolds's parametricity, respectively in its unary and binary variants. We exploit this correspondence to develop an original uniform indexed definition of both augmented semi-simplicial and semi-cubical sets, and fully formalize it in Coq.

1. Introduction

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

A family of sets can commonly be represented in two ways: as a family properly speaking, indexed over 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 set 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 (The Univalent Foundations Program, 2013) as the fibered/indexed equivalence[a].

$$\text{(fibered)} \qquad (\Sigma T : \mathsf{HSet}.(T \to S)) \simeq (S \to \mathsf{HSet}) \qquad \text{(indexed)}$$

where 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.

A *presheaf* on an category is a family of sets connected with maps indexed over the morphisms of the category. As such, it lives on the indexed side of the equivalence, contrasting with the fibered side, where we have *discrete Grothendieck fibrations* (Loregian and Riehl, 2020). However, there are situations where a presheaf F 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. This is the case in semi-simplicial and semi-cubical sets, where each $F(X)$ serving as source of a map $F(X) \overset{F(f)}{\to} F(Y)$ can be turned into a family $F(X)$ indexed on $F(Y)$. Since the maps are

---

[a]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 (The Univalent Foundations Program, 2013).

well-founded downward, this can be iterated from below, eventually producing a collection of families which we call an indexed presentation of the presheaf over the direct category.

Let us consider, for instance, the case of a semi-cubical set (Grandis and Mauri, 2003; Buchholtz and Morehouse, 2017) 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:

$$X_0 : \mathsf{HSet} \quad \begin{matrix} \leftarrow \partial^R - \\ \leftarrow \partial^L - \end{matrix} \quad X_1 : \mathsf{HSet} \quad \begin{matrix} \leftarrow \partial^{\star R} - \\ \leftarrow \partial^{\star L} - \\ \leftarrow \partial^{R\star} - \\ \leftarrow \partial^{L\star} - \end{matrix} \quad X_2 : \mathsf{HSet} \qquad \cdots$$

up to cubical faces identities. In it, X_1 can be seen as a family over $X_0 \times X_0$ presented in a fibered form, X_2 can be seen as a family of $X_1 \times X_1 \times X_1 \times X_1$, and so on. This suggests an alternative indexed presentation of the same presheaf as a stratified sequence of families indexed on elements of families of lower rank, taking into account the coherence conditions to prevent duplications. Formulated in type theory, it takes the form:

$$X_0 : \hspace{9cm} \mathsf{HSet}$$
$$X_1 : \hspace{7cm} X_0 \times X_0 \to \mathsf{HSet}$$
$$X_2 : \Pi abcd. \qquad X_1(a)(b) \times X_1(c)(d) \times X_1(a)(c) \times X_1(b)(d) \to \mathsf{HSet}$$
$$\cdots$$

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]. A couple of constructions were proposed since then. One, by Voevodsky (2012), relies on the presentation of semi-simplicial sets as a presheaf over increasing injective maps between finite ordinals. Another, by Herbelin (2015)[c], and fully formalized in the Coq proof assistant, relies on the presentation of semi-simplicial sets as a presheaf over face maps. Another by Part and Luo (2015), formalized in (an emulation of logic-enriched homotopy type theory in) the Plastic proof assistant, and yet another by Annenkov et al. (2017), formalized in (an emulation of a two-level type theory in) the Agda proof assistant [d], also 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, defining a concrete presheaf in indexed form only requires providing the family, since the responsibility of defining maps and showing the coherence conditions is already accounted for in the definition.

---

[b] See e.g. ncatlab.org/nlab/show/semi-simplicial+types+in+homotopy+type+theory.

[c] In hindsight, the title of the paper "A dependently-typed construction of semi-simplicial types" is somewhat confusing: it was part of a project for defining semi-simplicial types, but, in practice, the definition is done in a type theory with Unicity of Identity Proofs, meaning, in a type theory where types (intended to represent arbitrary ∞-groupoids) are actually sets (0-truncated ∞-groupoids or 0-groupoids). The confusion between the question of moving from semi-simplicial sets to semi-simplicial types, and moving from a presheaf to an indexed presentation was, however, common at the time.

[d] github.com/nicolaikraus/HoTT-Agda/blob/master/nicolai/SemiSimp/SStypes.agda

In category-theoretic terms, the indexing set of the family X_n in the indexed presentation is a *matching object* of the presheaf at n. The indexed presentation of presheaves over a direct category can then be identified with a subclass of ordinary presheaves known as Reedy fibrant, where Reedy fibrancy precisely expresses that the set X_n in the fibered presentation is a Σ-type over a representative of the class of equivalence of matching objects at level n (Shulman, 2014; Kraus and Sattler, 2017).

Reynolds' parametricity and its unary and binary variants

In the context of functional programming, Reynolds' parametricity (1983) interprets types as relations characterizing the observational behavior of programs of this type. More generally, *correspondences*, in the sense of "relevant" relation, that is of a family over a product of sets, can be used in place of relations. Parametricity can then be iterated, and relying on the fibered representation of correspondences as spans, it has been noted that iterated Reynolds' parametricity has the structure of a cubical set (Johann and Sojakova, 2017; Altenkirch and Kaposi, 2015; Moulin, 2016; Moeneclaey, 2021, 2022). We obtain a *unary* variant of Reynolds' *binary* parametricity by using predicates or families instead of relations or correspondences, in which case, we obtain a form of realizability (Bernardy and Moulin, 2012; Lasson, 2014; Moulin, 2016). It has then been noted that iterated unary parametricity has the structure of an augmented simplicial set [e]. This similarity 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 ν-sets, of presheaves over a ν-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 in full, 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 *ν-sets*, of augmented semi-simplicial and semi-cubical sets. Alternatively, from a category-theory perspective, we are characterizing, among the class of matching objects up to isomorphism for augmented semi-simplicial and cubical sets, a particular instance that reflects the structure of iterated parametricity.

This new construction is useful in the direction of better understanding the technicality of coherence issues in defining "semi-simplicial types" (Part and Luo, 2015; Shulman, 2014; Altenkirch et al., 2016; Kraus and Sattler, 2017), as well as the dialectic between reasoning in Extensional Type Theory and Intensional Type Theory.

Our work is a step in the direction of the program initiated in Altenkirch and Kaposi (2015) to develop parametricity-based models of parametric type theory (Bernardy et al., 2015; Nuyts et al., 2017; Cavallo and Harper, 2020) and cubical type theory (Bezem et al., 2013; Cohen et al., 2018; Angiuli et al., 2021) which are closer to the syntax of type theory, and is likely to better reflect the definitional properties of type theory than presheaf-based cubical sets would do. Compare, for example, the loss of definitional properties when interpreting "indexed" dependent types of type theory as "fibrations" in models such as locally cartesian closed categories (Curien et al., 2014).

Our mechanization can be found at github.com/artagnon/bonak. The construction was conceived in Summer 2019, and the mechanization began in late 2019. A sketch of

---

[e]Private communication with Hugo Moeneclaey and Thorsten Altenkirch.

the construction was presented at the 2020 HoTT-UF workshop and the completion of the mechanization was reported at the TYPES 2022 conference.

2. Semi-simplicial and semi-cubical sets

In this section, we generalize semi-simplicial and semi-cubical sets to ν-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 one 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. We can then draw augmented semi-simplicial sets like semi-simplicial sets, except for this shift by one.

While ordinary semi-simplicial sets are presheaves over the semi-simplex category, augmented semi-simplicial sets are presheaves over, what we will define as, Δ_+. There are different ways to define Δ_+, up to equivalence, and we use a definition that can be extended to semi-cubical sets in a straightforward manner.

Notation 1 (Finite sequences). *We denote finite sequences by $[i_1, \ldots i_n]$ for i_j ranging over some domain. In particular, the empty sequence is written $[]$ and we define $i :: [j_1, \ldots, j_n]$ to be $[i, j_1, \ldots, j_n]$.*

Definition 2 (Δ_+). *The definition of Δ_+ 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.*

$$
\begin{aligned}
\mathsf{Obj}_{\Delta_+} &:= \mathbb{N} \\
\mathsf{Hom}_{\Delta_+}(p, n) &:= \{l \in \{0, \star\}^n \mid \text{number of } \star \text{ in } l = p\} \\
g \circ f &:= \begin{cases} f & \text{if } g = [] \\ 0 :: (g' \circ f) & \text{if } g = (0 :: g') \\ a :: (g' \circ f') & \text{if } g = (\star :: g'), f = (a :: f'), \text{where } a = 0 \text{ or } \star \end{cases} \\
\mathsf{id} &:= [\star, \ldots, \star] \; n \text{ times for } \mathsf{id} \in \mathsf{Hom}_{\Delta_+}(n, n)
\end{aligned}
$$

Definition 3 (Set_{Δ_+}). *We define the category of augmented semi-simplicial sets as the functor category:*

$$
\mathsf{Set}_{\Delta_+} := \mathsf{Set}^{\Delta_+^{\mathrm{op}}}
$$

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

Definition 4 (Δ_+^n). *The standard augmented n-semi-simplex Δ_+^n is defined as the Yoneda embedding of $n \in \mathsf{Obj}(\Delta_+)$:*

$$\Delta_+^n : \mathsf{Set}_{\Delta_+}$$
$$\Delta_+^n(p) := \mathsf{Hom}(p, n)$$
$$\Delta_+^n(f) := \lambda g. \, g \circ f$$

The standard augmented 0-semi-simplex is a singleton made of one color (in our 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.
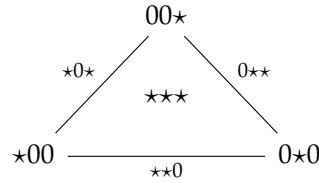
Example 5 (Δ_+^1). The standard augmented 1-semi-simplex can be pictured as a point, colored black, corresponding to the unique morphism in $\mathsf{Hom}(0, 1)$. This point is the identity in $\mathsf{Hom}(1, 1)$; it is hence shown as a singleton $\star$.

$$\star$$

Example 6 (Δ_+^2). The standard augmented 2-semi-simplex is drawn as two points, given by $\mathsf{Hom}(1, 2)$, along with a line connecting them, given by $\mathsf{Hom}(2, 2)$. We use the color black to denote the unique morphisms in $\mathsf{Hom}(0, 1)$ and $\mathsf{Hom}(0, 2)$.

$$\star 0 \xrightarrow{\ \star\star\ } 0\star$$

Example 7 (Δ_+^3). Δ_+^3 is drawn as three points, given by $\mathsf{Hom}(1, 3)$, three lines connecting them, given by $\mathsf{Hom}(2, 3)$, and a triangular filler given by $\mathsf{Hom}(3, 3)$.

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 $\star$. 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 $\star$ of the n-dimensional component with 0.

Semi-cubical sets

Semi-cubical sets are defined like augmented semi-simplicial sets except that Δ_+ is replaced by $\square$ in which we take sequences of L, R and $\star$, instead of sequences of 0 and $\star$. They represent ordinary semi-cubical sets, with faces (and no augmentation).

Definition 8 (□). *The definition of □ is shown below. The symbols L and R indicate opposite faces of a cube.*

$$\mathsf{Obj}_\square := \mathbb{N}$$
$$\mathsf{Hom}_\square(p, n) := \{l \in \{L, R, \star\}^n \mid \textit{number of } \star \textit{ in } l = p\}$$
$$g \circ f := \begin{cases} f & \textit{if } g = [] \\ a :: (g' \circ f) & \textit{if } g = (a :: g'), \textit{where } a = L \textit{ or } R \\ a :: (g' \circ f') & \textit{if } g = (\star :: g'), f = (a :: f'), \textit{where } a = L, R, \textit{ or } \star \end{cases}$$
$$\mathsf{id} := [\star, \dots, \star] \; n \textit{ times}$$

Again, 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.*

Definition 9 (Set$_\square$). *We define the category of semi-cubical sets as the functor category:*

$$\mathsf{Set}_\square := \mathsf{Set}^{\square^{\mathrm{op}}}$$

Definition 10 (□^{n}). *The standard semi-cube □^{n} is defined as the Yoneda embedding of $n \in$ Obj(□):*

$$\square^n : \mathsf{Set}_\square$$
$$\square^n(p) := \mathsf{Hom}(p, n)$$
$$\square^n(f) := \lambda g. \, g \circ f$$

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

Example 11 (□^{0}). □^{0} is Hom$(0, 0)$, or the singleton set of the empty sequence:

$$\{\}$$

Example 12 (□^{1}). □^{1} consists of two points, given by Hom$(0, 1)$, and a line, given by Hom$(1, 1)$.

$$L \xrightarrow{\quad \star \quad} R$$

Example 13 (□^{2}). □^{2} consists of four points, given by Hom$(0, 2)$, four lines connecting the four points, given by Hom$(1, 2)$, and a filler, given by Hom$(2, 2)$:

$$\begin{array}{ccc} LR & \xrightarrow{\;\star R\;} & RR \\ {\scriptstyle L\star}\Big\downarrow & \star\star & \Big\downarrow{\scriptstyle R\star} \\ LL & \xrightarrow[\;\star L\;]{} & RL \end{array}$$

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 $\star$. 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 $\star$ with L to the one obtained by replacing the leftmost $\star$ with R.

ν-**sets**

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

Cardinal of ν	1	2
Interpretation	Augmented semi-simplicial types	Semi-cubical types

To obtain this, we extend $\mathbf{\Delta}_+$ and $\square$ in a straightforward manner into a category which we call $\circ$.

Definition 14 ($\circ$). *The definition of ν-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.*

$$\mathsf{Obj}_{\circ} := \mathbb{N}$$
$$\mathsf{Hom}_{\circ}(p, n) := \{l \in (\nu \cup \{\star\})^n \mid \textit{number of } \star \textit{ in } l = p\}$$
$$g \circ f := \begin{cases} f & \textit{if } g = [] \\ a :: (g' \circ f) & \textit{if } g = (a :: g'), \textit{where } a \in \nu \\ a :: (g' \circ f') & \textit{if } g = (\star :: g'), f = (a :: f'), \textit{where } a \in \nu \textit{ or } a = \star \end{cases}$$
$$\mathsf{id} := [\star, \dots, \star] \; n \textit{ times for } \mathsf{id} \in \mathsf{Hom}_{\circ}(n, n)$$

A ν-set is thus a contravariant functor ϕ from the ν-semi-shape category to Set and we call n-ν-semi-shape an element of $\phi(n)$. As in the augmented semi-simplicial and semi-cubical cases, the standard $(n+1)$-ν-semi-shape is obtained by connecting together ν copies of the standard n-ν-semi-shape with an extra copy stretched in the new dimension. We clarify in the next sections how this process of construction is similar to the parametricity translation developed for functional programming (Reynolds, 1983) and more generally for type theory (Bernardy et al., 2010; Bernardy and Lasson, 2011; Atkey et al., 2014; Bernardy et al., 2015).

3. Type theory

Martin-Löf's Type theory (1975; 1984) 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 Agda (2023), Coq (2023) or Lean (2015). In type theory, propositions are types and proofs are programs. A particularity of type theory is also that types and programs,

hence propositions and proofs also, are considered modulo an equational theory called definitional equality.

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 (Kapulkin and Lumsdaine, 2021). These models support the univalence principle stating that equality of types mimics equivalence of types, leading to the development of Homotopy Type Theory (The Univalent Foundations Program, 2013). In type theory, types are organized in a hierarchy of universes written Type_m for m a natural number. Main types in type theory are the types of dependent pairs, written $\Sigma a : A.\, B(a)$, the types of dependent functions, written $\Pi a : A.\, B(a)$, for A a type and $B(a)$ a type dependent on the inhabitant a of A, and the type of propositional equality, written $t = u$. As a notation, the type of dependent pairs when B is actually 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 $A \to B$. We assume our type theory to also include a distinguished singleton type, written unit, and with inhabitant $*$, the type of boolean values, called bool, and the type of natural numbers. We also assume that our type theory includes the coinductive type of infinite tuples. We also write hd and tl the projections of dependent pairs, and refl for reflexivity. Logical propositions being types themselves, we use Π to represent universal quantification and Σ to represent existential quantification.

A type-theoretic notion of sets can be recovered in each universe as HSet_m, denoting the subtype of Type_m for which paths are degenerated, what can be expressed by the property of Uniqueness of Identity Proofs (UIP). Technically, this is expressed as a structure equipping a domain Dom with the property UIP:

$$\mathsf{Dom} : \mathsf{Type}_m$$
$$\mathsf{UIP} : \Pi xy : \mathsf{Dom}.\, \Pi pq : x = y.\, p = q$$

In HSet_m, the following properties hold:

(1) UIP holds on the unit type, bool type, as well as all types of finite cardinal ν.
(2) UIP propagates to Σ-types.
(3) UIP propagates to Π-types, with some additional functional extensionality axioms.

By the notations Type and HSet will be meant Type_m or HSet_m at some unspecified universe level m.

We are also interested in *extensional* type theory; i.e. in a type theory with the following reflection rule, where $=$ is propositional equality in some type and $\equiv$ is definitional equality (Martin-Löf, 1984):

$$\frac{\Gamma \vdash p : t = u}{\Gamma \vdash t \equiv u}$$

Note that the reflection rule implies UIP so that HSet and Type ae equivalent in extensional type theory. The reflection rules also implies functional extensionality, and, actually, extensional type theory is logically equivalent to intensional type theory extended with UIP and functional extensionality (Hofmann, 1995).

4. Relating to parametricity

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

$$X_0 : \hspace{10em} \mathsf{HSet}$$
$$X_1 : \hspace{6em} X_0 \times X_0 \to \mathsf{HSet}$$
$$X_2 : \Pi abcd. \hspace{2em} X_1(a)(b) \times X_1(c)(d) \times X_1(a)(c) \times X_1(b)(d) \to \mathsf{HSet}$$
$$\ldots$$

In there, the process of construction of the type of X_1 from that of X_0, and from the type of X_2 to that of X_1, and so on, is similar to iteratively applying a binary parametricity translation. The binary parametricity which we consider interprets a closed type A by a family $A_\star$ over $A \times A$, what can be seen as a graph whose vertices are in A. To each type constructor is thus associated the construction of a graph. To start with, the type of types HSet is interpreted as the family of type of families $\mathsf{HSet}_\star$, which takes A_L and A_R in HSet and returns the type $A_L \times A_R \to \mathsf{HSet}$ of families over A_L and A_R. Also, for A interpreted by $A_\star$ and B interpreted by $B_\star$, a dependent function type $\Pi a : A.\, B$ is interpreted as the graph $(\Pi a : A.\, B)_\star$ that takes two functions f_L and f_R of type $\Pi a : A.\, B$, and expresses that these functions map related arguments in A to related arguments in B:

$$(\Pi a : A.\, B)_\star(f_L, f_R) \triangleq \Pi(a_L, a_R) : (A \times A).\, (A_\star(a_L, a_R) \to B_\star(f_L(a_L), f_R(a_R)))$$

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

$$(A \times B)_\star((a_L, b_L), (a_R, b_R)) \triangleq A_\star(a_L, a_R) \times B_\star(b_L, b_R)$$

In particular, for $X : \mathsf{HSet}$, applying our parametricity translation is about associating to X an inhabitant $X_\star$ of $\mathsf{HSet}_\star(X, X)$ i.e. of $X \times X \to \mathsf{HSet}$. In turn, applying the translation again to $X_\star : X \times X \to \mathsf{HSet}$ is about associating to $X_\star$ an inhabitant $X_{\star\star}$ of $(X \times X \to \mathsf{HSet})_\star(X_\star, X_\star)$ i.e. of:

$$\Pi((x_{LL}, x_{lR}), (x_{RL}, x_{RR})) : ((X \times X) \times (X \times X)).$$
$$(X_\star(x_{LL}, x_{LR}) \times X_\star(x_{RL}, x_{RR}) \to X_\star(x_{LL}, x_{RL}) \times X_\star(x_{LR}, x_{RR}) \to \mathsf{HSet})$$

which hints us at 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 $X : \mathsf{HSet}$:

$$X_0 \triangleq X \quad : \hspace{6em} \mathsf{HSet}$$
$$X_1 \triangleq X_\star \quad : \hspace{4em} \mathsf{HSet}_\star(X, X)$$
$$X_2 \triangleq X_{\star\star} : (\mathsf{HSet}_\star(X, X))_\star(X_\star, X_\star)$$

$$\ldots$$

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, $X_{n+1} : (S_n)_\star(X_n, X_n)$ 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 ν-sets.

5. Our construction

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

Sections 5.1 and 5.3 describe the construction at an informal level of discourse:

(1) In 5.1, we present it in informal extensional type theory where equational reasoning is left implicit.
(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 5.3 thus rephrases the construction in (informal) intensional type theory.

Sections 5.4, 5.5, and 5.6 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 5.4.
(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 5.5.
(3) A 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 5.6.

5.1 The construction in informal type theory

A ν-set is a sequence of families of HSet, that is HSet_m for some universe level m. We call such sequence a ν-set at level m, whose type thus lives in HSet_{m+1}.

Table 1 describes the type of a ν-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 ν-sets: $\nu\mathsf{Set}_m^{\geq n}$ denotes an infinite sequence $X_{n+1}, X_{n+2}, \ldots$ dependent on a n-truncated ν-set, $\nu\mathsf{Set}_m^{<n}$, so that, when m is 0, it denotes a full ν-set, written $\nu\mathsf{Set}_m$. This is made possible because the 0-truncated ν-set, $\nu\mathsf{Set}_m^{<0}$, is degenerated: it is an empty family, defined to be the unit type, and there is thus only one 0-truncated ν-set, namely the canonical inhabitant $\star$ of unit.

The definition of the type of a n-truncated ν-set is in turn described in table 2. In the infinite sequence of type families representing a ν-set, the n-th component is a type

dependent over a fullframe. The type fullframe defines a matching object. It is recursively defined in table 3, using the auxiliary definitions of frame, layer and painting. The rationale behind frame, layer and painting will be explained later but it should already be noticed that the type layer relies on an operator of frame restriction $\mathrm{restr}_{\mathrm{frame}}$ which is defined in table 4, and this restriction operator is in turn defined using auxiliary definitions $\mathrm{restr}_{\mathrm{layer}}$ and $\mathrm{restr}_{\mathrm{painting}}$.

Notably, the definition of $\mathrm{restr}_{\mathrm{layer}}$ relies on an equality expressing the commutation of the composition of $\mathrm{restr}_{\mathrm{frame}}$. 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 frame, layer, and painting. This is what $\mathrm{coh}_{\mathrm{frame}}$ does using auxiliary proofs $\mathrm{coh}_{\mathrm{layer}}$ and $\mathrm{coh}_{\mathrm{painting}}$. Even though it looks independent of the definitions from the the other tables, $\mathrm{coh}_{\mathrm{frame}}$ has to be proved mutually with the definitions of frame, layer, painting, and their corresponding restrictions. More precisely, for a fixed n, the block of frame, $\mathrm{restr}_{\mathrm{frame}}$, and $\mathrm{coh}_{\mathrm{frame}}$ has to be defined in one go by induction on p. Also, each of painting, $\mathrm{restr}_{\mathrm{painting}}$, and $\mathrm{coh}_{\mathrm{painting}}$ is built by induction from p to n. The painting block at n relies on the frame block at n, but the converse dependency is only on lower n, so this is well-founded. Note that layer, $\mathrm{restr}_{\mathrm{layer}}$ and $\mathrm{coh}_{\mathrm{layer}}$ are just abbreviations. The exact way this mutual recursion is eventually formalized is explained in section 5.4.

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.

$\nu\mathrm{Set}_m$		:	HSet_{m+1}
$\nu\mathrm{Set}_m$		$\triangleq$	$\nu\mathrm{Set}_m^{\geq 0}(*)$
$\nu\mathrm{Set}_m^{\geq n}$	$(D:\nu\mathrm{Set}_m^{<n})$	:	HSet_{m+1}
$\nu\mathrm{Set}_m^{\geq n}$	D	$\triangleq$	$\Sigma R:\nu\mathrm{Set}_m^{=n}(D).\nu\mathrm{Set}_m^{\geq n+1}(D,R)$

Table 1. Main definition

$\nu\mathrm{Set}_m^{<n}$		:	HSet_{m+1}
$\nu\mathrm{Set}_m^{<0}$		$\triangleq$	unit
$\nu\mathrm{Set}_m^{<n'+1}$		$\triangleq$	$\Sigma D:\nu\mathrm{Set}_m^{<n'}.\nu\mathrm{Set}_m^{=n'}(D)$
$\nu\mathrm{Set}_m^{=n}$	$(D:\nu\mathrm{Set}_m^{<n})$	:	HSet_m
$\nu\mathrm{Set}_m^{=n}$	D	$\triangleq$	$\mathrm{fullframe}_m^n(D)\to\mathrm{HSet}_m$

Table 2. Truncated ν-sets, the core

$\mathsf{fullframe}^n$	$(D : \nu\mathsf{Set}_m^{<n})$	:	HSet_m
$\mathsf{fullframe}^n$	D	≜	$\mathsf{frame}^{n,n}(D)$
$\mathsf{frame}^{n,p,p\leq n}$	$(D : \nu\mathsf{Set}_m^{<n})$	:	HSet_m
$\mathsf{frame}^{n,0}$	D	≜	unit
$\mathsf{frame}^{n,p'+1}$	D	≜	$\Sigma d : \mathsf{frame}^{n,p'}(D).\,\mathsf{layer}^{n,p'}(d)$
$\mathsf{layer}^{n,p,p<n}$	$\{D : \nu\mathsf{Set}_m^{<n}\}$ $(d : \mathsf{frame}^{n,p}(D))$	:	HSet_m
$\mathsf{layer}^{n,p}$	$D\,d$	≜	$\Pi\omega.\,\mathsf{painting}^{n-1,p}(D.2)(\mathsf{restr}^{n,p}_{\mathsf{frame},\omega,p}(d))$
$\mathsf{painting}^{n,p,p\leq n}$	$(D : \nu\mathsf{Set}_m^{<n})$ $(E : \nu\mathsf{Set}_m^{=n}(D))$ $(d : \mathsf{frame}^{n,p}(D))$	:	HSet_m
$\mathsf{painting}^{n,p,p=n}$	$D\,E\,d$	≜	$E(d)$
$\mathsf{painting}^{n,p,p<n}$	$D\,E\,d$	≜	$\Sigma l : \mathsf{layer}^{n,p}(d).\,\mathsf{painting}^{n,p+1}(E)(d,l)$

Table 3. frame, layer, and painting

$\mathsf{restr}^{n,p,p\leq q\leq n-1}_{\mathsf{frame},\epsilon,q}$	$\{D : \nu\mathsf{Set}^{<n}\}$ $(d : \mathsf{frame}^{n,p}(D))$	:	$\mathsf{frame}^{n-1,p}(D.1)$
$\mathsf{restr}^{n,0}_{\mathsf{frame},\epsilon,q}$	$D\,*$	≜	$*$
$\mathsf{restr}^{n,p'+1}_{\mathsf{frame},\epsilon,q}$	$D\,(d,l)$	≜	$(\mathsf{restr}^{n,p'}_{\mathsf{frame},\epsilon,q}(d),\ \mathsf{restr}^{n,p'}_{\mathsf{layer},\epsilon,q-1}(l))$
$\mathsf{restr}^{n,p,p\leq q\leq n-2}_{\mathsf{layer},\epsilon,q}$	$\{D : \nu\mathsf{Set}^{<n}\}$ $\{d : \mathsf{frame}^{n,p}(D)\}$ $(l : \mathsf{layer}^{n,p}(d))$	:	$\mathsf{layer}^{n-1,p}(\mathsf{restr}^{n,p}_{\mathsf{frame},\epsilon,q+1}(d))$
$\mathsf{restr}^{n,p}_{\mathsf{layer},\epsilon,q}$	$D\,d\,l$	≜	$\lambda\omega.(\mathsf{restr}^{n-1,p}_{\mathsf{painting},\epsilon,q}(D.2)(l_\omega))$
$\mathsf{restr}^{n,p,p\leq q\leq n-1}_{\mathsf{painting},\epsilon,q}$	$(D : \nu\mathsf{Set}^{<n})$ $(E : \nu\mathsf{Set}^{=n}(D))$ $(d : \mathsf{frame}^{n,p}(D))$ $(c : \mathsf{painting}^{n,p}(E)(d))$	:	$\mathsf{painting}^{n-1,p}(D.2)(\mathsf{restr}^{n,p}_{\mathsf{frame},\epsilon,q+1}(d))$
$\mathsf{restr}^{n,p,p=q}_{\mathsf{painting},\epsilon,q}$	$D\,E\,d\,(l,\_)$	≜	l_ϵ
$\mathsf{restr}^{n,p,p<q}_{\mathsf{painting},\epsilon,q}$	$D\,E\,d\,(l,c)$	≜	$(\mathsf{restr}^{n,p}_{\mathsf{layer},\epsilon,q}(l),\ \mathsf{restr}^{n,p+1}_{\mathsf{painting},\epsilon,q}(E)(c))$

Table 4. q-th projection of restr, or faces

$\mathsf{coh}^{n,p,p\leq r\leq q\leq n-2}_{\mathsf{frame},\epsilon,\omega,q,r}$	$\{D : \nu\mathsf{Set}^{<n}\}$ $(d : \mathsf{frame}(D))$	:	$\mathsf{restr}^{n-1,p}_{\mathsf{frame},\epsilon,q}(\mathsf{restr}^{n,p}_{\mathsf{frame},\omega,r}(d))$ $= \mathsf{restr}^{n-1,p}_{\mathsf{frame},\omega,r}(\mathsf{restr}^{n,p}_{\mathsf{frame},\epsilon,q+1}(d))$
$\mathsf{coh}^{n,0}_{\mathsf{frame},\epsilon,\omega,q,r}$	$D\,*$	≜	$\mathsf{refl}(*)$
$\mathsf{coh}^{n,p'+1}_{\mathsf{frame},\epsilon,\omega,q,r}$	$D\,(d,l)$	≜	$(\mathsf{coh}^{n,p'}_{\mathsf{frame},\epsilon,\omega,q,r}(d),\ \mathsf{coh}^{n,p'}_{\mathsf{layer},\epsilon,\omega,q,r}(l))$
$\mathsf{coh}^{n,p,p<r\leq q\leq n-2}_{\mathsf{layer},\epsilon,\omega,q,r}$	$(D : \nu\mathsf{Set}^{<n})$ $\{d : \mathsf{frame}(D)\}$ $(l : \mathsf{layer}(d))$	:	$\mathsf{restr}^{n-1,p}_{\mathsf{layer},\epsilon,q}(\mathsf{restr}^{n,p}_{\mathsf{layer},\omega,r}(l))$ $= \mathsf{restr}^{n-1,p}_{\mathsf{layer},\omega,r}(\mathsf{restr}^{n,p}_{\mathsf{layer},\epsilon,q+1}(l))$
$\mathsf{coh}^{n,p}_{\mathsf{layer},\epsilon,\omega,q,r}$	$D\,d\,l$	≜	$\lambda\theta.\,\mathsf{coh}^{n-1,p}_{\mathsf{painting},\epsilon,\omega,q-1,r-1}(D.2)(l_\theta)$
$\mathsf{coh}^{n,p,p\leq r\leq q\leq n-2}_{\mathsf{painting},\epsilon,\omega,q,r}$	$(D : \nu\mathsf{Set}^{<n})$ $(E : \nu\mathsf{Set}^{=n}(D))$ $(d : \mathsf{frame}(D))$ $(c : \mathsf{painting}(E)(d))$	:	$\mathsf{restr}^{n-1,p}_{\mathsf{painting},\epsilon,q}(D.2)(\mathsf{restr}^{n,p}_{\mathsf{painting},\omega,r}(E)(c))$ $= \mathsf{restr}^{n-1,p}_{\mathsf{painting},\omega,r}(D.2)(\mathsf{restr}^{n,p}_{\mathsf{painting},\epsilon,q+1}(E)(c))$
$\mathsf{coh}^{n,p,p=r}_{\mathsf{painting},\epsilon,\omega,q,r}$	$D\,E\,d\,(l,\_)$	≜	$\mathsf{refl}(\mathsf{restr}^{n-1,p}_{\mathsf{painting},\epsilon,q-1}(D.2)(l_\epsilon))$
$\mathsf{coh}^{n,p,p<r}_{\mathsf{painting},\epsilon,\omega,q,r}$	$D\,E\,d\,(l,c)$	≜	$(\mathsf{coh}^{n,p}_{\mathsf{layer},\epsilon,\omega,q,r}(l),\ \mathsf{coh}^{n,p+1}_{\mathsf{painting},\epsilon,\omega,q,r}(E)(c))$

Table 5. Commutation of q-th projection and r-th projection, or coherence conditions

5.2 Intuition for our formal construction

To assign types to $X_0, X_1, X_2, \ldots$ in the indexed representation, we give a recursive definition relying on the building blocks called frame, layer, and painting. A frame is a boundary of a standard form (simplex, cube), which we decompose into layer, and a painting corresponds to a filled frame. Some frame are full and we call them fullframe: they represent matching objects.

Every X_n is uniformly assigned a type of the form $\mathsf{fullframe}^n \to \mathsf{HSet}$. Compared to the informal description given in section 4, this amounts to applying isomorphisms such as $(A \to B \to C) \simeq (A \times B \to C)$ or $(\Pi a : A.\,(Ba \to C)) \simeq ((\Sigma a : A.\,Ba) \to C)$. In particular, $\mathsf{fullframe}^n$ is a "telescope", or a nesting of Σ-types.

We now illustrate how to recursively build $\mathsf{fullframe}^n$. To begin, let us set $\mathsf{fullframe}^0 = \mathsf{unit}$, so that the type HSet given to X_0 in section 4 can be equivalently formulated as $\mathsf{unit} \to \mathsf{HSet}$. Then, more generally, let each $\mathsf{fullframe}^n$ consist of n layers, written $\mathsf{layer}^{n,p}$ with $p < n$, that we stack in order, starting from unit, and writing $\mathsf{frame}^{n,p}$ for the p first layers of a $\mathsf{fullframe}^n$, so that $\mathsf{fullframe}^n$ is $\mathsf{frame}^{n,n}$. For instance, X_1 is made of one layer, so that it can be written as a Σ-type of an inhabitant of unit and $\mathsf{layer}^{1,0}$, as shown below. The figure also mentions how the type of X_2 is structured.

Let us now illustrate the construction of $\mathsf{fullframe}^3$, necessary to build the type of X_3.
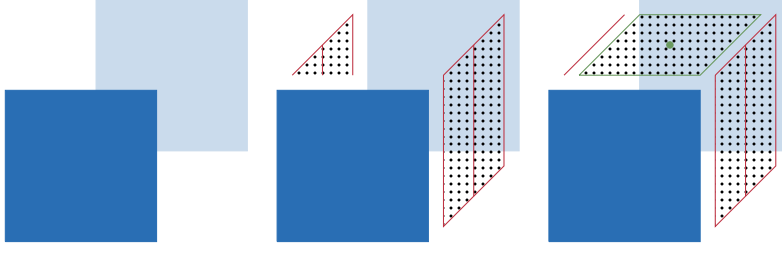
The figure on the left is $\mathsf{frame}^{3,1}$, in the middle is $\mathsf{frame}^{3,2}$, and on the right is $\mathsf{frame}^{3,3}$, which is full. Further, $\mathsf{frame}^{3,1}$ is made of one layer, $\mathsf{layer}^{3,0}$, shown in blue, $\mathsf{frame}^{3,2}$ is made of one additional layer, $\mathsf{layer}^{3,1}$, shown in red, $\mathsf{frame}^{3,3}$ is made of one more layer, $\mathsf{layer}^{3,2}$, shown in green.

We illustrated here the cubical case, that is $\nu = 2$, but, in general, a $\mathsf{layer}^{n,p}$ is a product of ν $\mathsf{painting}^{n-1,p}$. A $\mathsf{painting}^{n,0}$ is a n-dimensional object corresponding to a filled $\mathsf{fullframe}^n$. More generally, a $\mathsf{painting}^{n,p}$ is an n-dimensional object which has the form of a $\mathsf{painting}^{n-p,0}$, thus of $(n-p)$-dimensional form, but shifted and living in dimensions p to n. Such $\mathsf{painting}^{n,p}$ fills a space framed by a partial $\mathsf{frame}^{n,p}$ so that, together, they form a filled $\mathsf{fullframe}^n$. For instance, in the picture, each of the two $\mathsf{painting}^{2,0}$ of $\mathsf{layer}^{3,0}$ is a filled blue square, each of the two $\mathsf{painting}^{2,1}$ of $\mathsf{layer}^{3,1}$ is the 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 $\mathsf{painting}^{2,2}$ of $\mathsf{layer}^{3,2}$ is a 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. A $\mathsf{painting}^{n,p}$ complements a $\mathsf{frame}^{n,p}$ by adding layers needed to form a $\mathsf{fullframe}^n$ and by filling the resulting $\mathsf{fullframe}^n$ with an inhabitant of X_n. Layers are added from dimension n to dimension p that is opposite to the order from 0 to p the $\mathsf{frame}^{n,p}$ are built, as shown below.

$$
\begin{aligned}
\mathsf{frame}^{n,p} &\triangleq \Sigma a_n : (\dots (\Sigma * : \mathsf{unit}.\, \mathsf{layer}^{n,0})\dots).\, \mathsf{layer}^{n,p-1} \\
\mathsf{painting}^{n,p} &\triangleq \Sigma l_p : \mathsf{layer}^{n,p}.(\dots (\Sigma l_n : \mathsf{layer}^{n,n-1}.\, X_n)\dots)
\end{aligned}
$$

So far, we have not paid attention to the fact that we have a dependent type, shown as a Σ-type. Let us be more precise about this requirement. First, $\mathsf{fullframe}^n$ depends on all X_i up to $n-1$. So, we need to package up X_i, for $i < n$, into a nesting of Σ-types, constituting the type of a n-truncated ν-set, which we wrote $\nu\mathsf{Set}^{<n}$. This allows us to give the type $\nu\mathsf{Set}^{<n} \to \mathsf{HSet}$ to $\mathsf{fullframe}^n$. Then, for $D : \nu\mathsf{Set}^{<n}$, representing an initial prefix of $X_0, X_1, \dots X_{n-1}$, the indexed set X_n has type $\mathsf{fullframe}^n(D) \to \mathsf{HSet}$. Thus, $\mathsf{frame}^{n,p}$, $\mathsf{layer}^{n,p}$ and $\mathsf{painting}^{n,p}$ also depend on D. We can then reformulate the previous equation with its dependency on D. In particular, X_n is just the last component of D, that is $D.\mathsf{tl}$.

$$
\begin{aligned}
\mathsf{frame}^{n,p}(D) &\triangleq \Sigma a_n : (\dots (\Sigma * : \mathsf{unit}.\, \mathsf{layer}^{n,0}(D))\dots).\, \mathsf{layer}^{n,p-1}(D) \\
\mathsf{painting}^{n,p}(D) &\triangleq \Sigma l_p : \mathsf{layer}^{n,p}(D).\, (\dots (\Sigma l_n : \mathsf{layer}^{n,n-1}(D).\, D.\mathsf{tl})\dots)
\end{aligned}
$$

An 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 $\mathsf{layer}^{n,p}$ has to depend on $\mathsf{frame}^{n,p}$. We also need a way to characterize the ν borders of each $\mathsf{painting}^{n-1,p}$ that composes a $\mathsf{layer}^{n,p}$, and this is where the restriction $\mathsf{restr}^{n,p}_{\mathsf{frame},\epsilon,p}$ arrives, for each $\epsilon < \nu$.

For instance, on the picture, the left and right painting^{2,1}, shown in red, are laid on respectively the left and right borders of the blue boxes, and hence needs to depend on frame^{3,1}. The left and right borders of the two blue boxes are extracted as restr$^{2,1}_{\text{frame},L}(D)(d)$ and restr$^{2,1}_{\text{frame},R}(D)(d)$. We can then refine again the previous equation by showing the dependencies on d, as shown below.

$$\text{frame}^{n,p}(D) \triangleq \Sigma d : (\ldots (\Sigma * : \text{unit}. \text{layer}^{n,0}(D)(*)) \ldots). \text{layer}^{n,p}(D)(d)$$

$$\text{painting}^{n,p}(D)(d) \triangleq \Sigma l_p : \text{layer}^{n,p}(D)(d). (\ldots (\Sigma l_n : \text{layer}^{n,n-1}(D)(d, l_p, \ldots, l_{n-1}).$$

$$D.\text{tl}(d, l_p, \ldots, l_n)) \ldots)$$

$$\text{where } (d, l_p, \ldots, l_q) \text{ abbreviates } ((\ldots (d, l_p), \ldots), l_q)$$

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

$$\text{layer}^{n,p}(D)(d) \triangleq \text{painting}^{n-1,p}(D.\text{hd})\big(\text{restr}^{n,p}_{\text{frame},L,p}(d)\big)$$

$$\times$$

$$\text{painting}^{n-1,p}(D.\text{hd})\big(\text{restr}^{n,p}_{\text{frame},R,p}(d)\big)$$

The operation restr$^{n,p}_{\text{frame},\epsilon,q}$ restricts the p first layers of a frame, and the construction is by recursion on the structure of a frame d. This necessitates the definitions restr$^{n,p}_{\text{layer},\epsilon,q}(d)(l)$ and restr$^{n,p}_{\text{painting},\epsilon,q}(d)(c)$, for l a layer and c a painting. The key case is restr$^{n,p}_{\text{painting},\epsilon,p}(d)(c)$, where c, a painting^{n,p}, has necessarily the form of $((c_L, c_R), \_)$: restr$^{n,p}_{\text{painting},L,p}$ picks out c_L, a painting$^{n-1,p}$, restr$^{n,p}_{\text{painting},R,p}$ picks out the c_R, also a painting$^{n-1,p}$, and $\_$, a painting$^{n,p+1}$, is discarded. There is one last difficulty, which we illustrate by writing down expected and actual types.

Given c_ω of type

$$c_\omega : \text{painting}^{n-1,p}(D.\text{hd})\big(\text{restr}^{n-1,p}_{\text{frame},\omega,q}(d)\big)$$

restr$^{n,p}_{\text{layer},\epsilon}(d)(c_L, c_R)$ produces a layer in which the ω-component has the type

$$\text{painting}^{n-2,p}(D.\text{hd.hd})\big(\text{restr}^{n-1,p}_{\text{frame},\epsilon,q}(\text{restr}^{n,p}_{\text{frame},\omega,p}(d))\big)$$

while we expect a component of type

$$\text{painting}^{n-2,p}(D.\text{hd.hd})\big(\text{restr}^{n-1,p}_{\text{frame},\omega,p}(\text{restr}^{n,p}_{\text{frame},\epsilon,q+1}(d))\big)$$

Hence, we need a coherence condition to commute the restrictions. Coherence conditions similar to this necessitate what are shown as, coh$_{\text{frame}}$, coh$_{\text{layer}}$ and coh$_{\text{painting}}$ in table 5. These are by induction on the structure of frame, layer and painting. Note that, for the construction in intensional type theory, we further need a 2-dimensional coherence condition, coh2$_{\text{frame}}$, for coh$_{\text{layer}}$. This is explained in the next section.

5.3 From extensional to intensional type theory: making explicit the equality proofs

In this section, we intend to get rid of the reflection rule and make explicit what is needed to rephrase the construction in intensional type theory. For readability purpose, we however make explicit only 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 other steps are standard enough to be omitted at this stage. See section 5.6 for the details.

The need for transport along a proof of commutation of $\text{restr}_{\text{frame}}$ in the definition of $\text{restr}_{\text{layer}}$ is made explicit in table 4′, where the arrow over $\text{coh}_{\text{frame}}$ indicates the direction of rewrite.

$\text{restr}^{n,p,p\le q\le n-1}_{\text{frame},\epsilon,q}$	$\{D:\nu\text{Set}^{<n}\}$ $(d:\text{frame}^{n,p}(D))$	:	$\text{frame}^{n-1,p}(D.1)$
$\text{restr}^{n,0}_{\text{frame},\epsilon,q}$	$D *$	≜	$*$
$\text{restr}^{n,p'+1}_{\text{frame},\epsilon,q}$	$D\,(d,l)$	≜	$(\text{restr}^{n,p'}_{\text{frame},\epsilon,q}(d), \text{restr}^{n,p'}_{\text{layer},\epsilon,q-1}(l))$
$\text{restr}^{n,p,p\le q\le n-2}_{\text{layer},\epsilon,q}$	$\{D:\nu\text{Set}^{<n}\}$ $\{d:\text{frame}^{n,p}(D)\}$ $(l:\text{layer}^{n,p}(d))$	:	$\text{layer}^{n-1,p}(\text{restr}^{n,p}_{\text{frame},\epsilon,q+1}(d))$
$\text{restr}^{n,p}_{\text{layer},\epsilon,q}$	$D\,d\,l$	≜	$\lambda\omega.(\overrightarrow{\text{coh}^{n,p}_{\text{frame},\epsilon,\omega,q,p}}(d)(\text{restr}^{n-1,p}_{\text{painting},\epsilon,q}(D.2)(l_\omega)))$
$\text{restr}^{n,p,p\le q\le n-1}_{\text{painting},\epsilon,q}$	$(D:\nu\text{Set}^{<n})$ $(E:\nu\text{Set}^{=n}(D))$ $(d:\text{frame}^{n,p}(D))$ $(c:\text{painting}^{n,p}(E)(d))$	:	$\text{painting}^{n-1,p}(D.2)(\text{restr}^{n,p}_{\text{frame},\epsilon,q+1}(d))$
$\text{restr}^{n,p,p=q}_{\text{painting},\epsilon,q}$	$D\,E\,d\,(l,\_)$	≜	l_ϵ
$\text{restr}^{n,p,p<q}_{\text{painting},\epsilon,q}$	$D\,E\,d\,(l,c)$	≜	$(\text{restr}^{n,p}_{\text{layer},\epsilon,q}(l), \text{restr}^{n,p+1}_{\text{painting},\epsilon,q}(E)(c))$

Table 4′. q-th projection of restr, or faces

$\text{coh}^{n,p,p\le r\le q\le n-2}_{\text{frame},\epsilon,\omega,q,r}$	$\{D:\nu\text{Set}^{<n}\}$ $(d:\text{frame}^{n,p}(D))$	:	$\text{restr}^{n-1,p}_{\text{frame},\epsilon,q}(\text{restr}^{n,p}_{\text{frame},\omega,r}(d))$ $= \text{restr}^{n-1,p}_{\text{frame},\omega,r}(\text{restr}^{n,p}_{\text{frame},\epsilon,q+1}(d))$
$\text{coh}^{n,0}_{\text{frame},\epsilon,\omega,q,r}$	$D *$	≜	$\text{refl}(*)$
$\text{coh}^{n,p'+1}_{\text{frame},\epsilon,\omega,q,r}$	$D\,(d,l)$	≜	$(\text{coh}^{n,p'}_{\text{frame},\epsilon,\omega,q,r}(d), \text{coh}^{n,p'}_{\text{layer},\epsilon,\omega,q,r}(l))$
$\text{coh}^{n,p,p<r\le q\le n-2}_{\text{layer},\epsilon,\omega,q,r}$	$(D:\nu\text{Set}^{<n})$ $\{d:\text{frame}^{n,p}(D)\}$ $(l:\text{layer}^{n,p}(d))$	:	$\overrightarrow{\text{coh}^{n,p}_{\text{frame},\epsilon,\omega,q,r}}(d)(\text{restr}^{n-1,p}_{\text{layer},\epsilon,q}(\text{restr}^{n,p}_{\text{layer},\omega,r}(l)))$ $= \text{restr}^{n-1,p}_{\text{layer},\omega,r}(\text{restr}^{n,p}_{\text{layer},\epsilon,q+1}(l))$
$\text{coh}^{n,p}_{\text{layer},\epsilon,\omega,q,r}$	$D\,d\,l$	≜	$\lambda\theta.(\overrightarrow{\text{coh2}^{n,p}_{\text{frame},\epsilon,\omega,\theta,q,r}}(d))(\text{ap}(\overrightarrow{\text{coh}^{n-1,p}_{\text{frame},\omega,\theta,r,p}}(\text{restr}^{n,p}_{\text{frame},\epsilon,q+2}(d))))$ $(\text{ap}(\text{restr}^{n-1,p}_{\text{frame},\omega,r}(\overrightarrow{\text{coh}^{n,p}_{\text{frame},\epsilon,\theta,q+1,p}})))\,\text{coh}^{n-1,p}_{\text{painting},\epsilon,\omega,q-1,r-1}(D.2)(l_\theta)$
$\text{coh}^{n,p,p\le r\le q\le n-2}_{\text{painting},\epsilon,\omega,q,r}$	$(D:\nu\text{Set}^{<n})$ $(E:\nu\text{Set}^{=n}(D))$ $(d:\text{frame}^{n,p}(D))$ $(c:\text{painting}^{n,p}(E)(d))$	:	$\overrightarrow{\text{coh}^{n,p}_{\text{frame},\epsilon,\omega,q,r}}(d)(\text{restr}^{n-1,p}_{\text{painting},\epsilon,q}(D.2)(\text{restr}^{n,p}_{\text{painting},\omega,r}(E)(c)))$ $= \text{restr}^{n-1,p}_{\text{painting},\omega,r}(D.2)(\text{restr}^{n,p}_{\text{painting},\epsilon,q+1}(E)(c))$
$\text{coh}^{n,p,p=r}_{\text{painting},\epsilon,\omega,q,r}$	$D\,E\,d\,(l,\_)$	≜	$\text{refl}(\text{restr}^{n-1,p}_{\text{painting},\epsilon,q-1}(D.2)(l_\epsilon))$
$\text{coh}^{n,p,p<r}_{\text{painting},\epsilon,\omega,q,r}$	$D\,E\,d\,(l,c)$	≜	$(\text{coh}^{n,p}_{\text{layer},\epsilon,\omega,q,r}(l), \text{coh}^{n,p+1}_{\text{painting},\epsilon,\omega,q,r}(E)(c))$

Table 5′. Commutation of q-th projection and r-th projection, or coherence conditions

The proof of $\text{coh}_{\text{frame}}$ itself requires making explicit several rewritings which were invisible in extensional type theory. The commutation of $\text{restr}_{\text{layer}}$ lives in a type referring to $\text{coh}_{\text{frame}}$, so we need a transport along the commutation of $\text{restr}_{\text{frame}}$ in the statement of $\text{coh}_{\text{layer}}$. The proof of $\text{coh}_{\text{layer}}$ is the most involved proof of the construction. It requires a higher-dimensional coherence condition, $\text{coh2}_{\text{frame}}$, whose exact formulation

is as follows.

$$\text{coh}^{n,p}_{\text{frame},\omega,\theta,r,p}(\text{restr}^{n,p}_{\text{frame},\epsilon,q+2}(d)) \bullet$$
$$\text{ap restr}^{n,p}_{\text{frame},\omega,r}(\text{coh}^{n,p}_{\text{frame},\epsilon,\theta,q+1,p}(d)) \bullet$$
$$\text{coh}^{n,p}_{\text{frame},\epsilon,\omega,q,r}(\text{restr}^{n,p}_{\text{frame},\theta,p}(d)) =$$
$$\text{ap restr}^{n,p}_{\text{frame},\theta,p}(\text{coh}^{n,p}_{\text{frame},\epsilon,\omega,q+1,r+1}(d)) \bullet$$
$$\text{coh}^{n,p}_{\text{frame},\epsilon,\theta,q,p}(\text{restr}^{n,p}_{\text{frame},\omega,r+1}(d)) \bullet$$
$$\text{ap restr}^{n,p}_{\text{frame},\epsilon,q}(\text{coh}^{n,p}_{\text{frame},\omega,\theta,r,p}(d))$$

where ap applies a function on two sides of an equality, and $\bullet$ is transitivity of equality. This property of equality proofs holds in HSet, and since our formalization is done in HSet, the term is trivially discharged.

Notice that each $\text{restr}_{\text{layer}}$ in the type of $\text{coh}_{\text{layer}}$ is hiding a $\text{coh}_{\text{frame}}$ rewrite: this makes a sum total of three $\text{coh}_{\text{frame}}$ rewrites on the left-hand side, and two $\text{coh}_{\text{frame}}$ rewrites on the right-hand side. In the proof term of $\text{coh}_{\text{layer}}$, $\text{coh}_{\text{painting}}$ has one $\text{coh}_{\text{frame}}$ rewrite on its left-hand side and zero on the right-hand side. This, combined with the two terms of the form $\text{ap coh}_{\text{frame}}$, matches our expectation of three $\text{coh}_{\text{frame}}$ on the left-hand side and two $\text{coh}_{\text{frame}}$ on the right-hand side. Then, $\text{coh2}_{\text{frame}}$ can be seen as expressing the commutation of these $\text{coh}_{\text{frame}}$ terms.

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

If we were not working in HSet, but in HGpd we would need to prove one more higher-dimensional coherence, and if we were working in Type, we would need to prove arbitrarily many higher-dimensional coherences. Here, HGpd is the subset of types A such that for all x and y in A, $x = y$ is in HSet. See Herbelin (2015); Altenkirch et al. (2016); Kraus (2021) for a discussion on the need for recursive higher-dimensional coherence conditions in formulating higher-dimensional structures in type theory.

5.4 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: we 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 $\text{restr}^n_{\text{frame}}$ depends on frame^n and frame^{n-1}, while $\text{coh}^n_{\text{frame}}$ depends on frame^n, frame^{n-1}, and frame^{n-2}, we only need to keep track of three "levels", and we built separate data structures for the levels, with dependencies. More concretely, we build the ten definitions shown in the tables by induction, and this is part of the definition of a larger record. The other fields of the record are frame, layer, painting at levels $n-1$ and $n-2$, $\text{restr}_{\text{frame}}$, $\text{restr}_{\text{layer}}$, and $\text{restr}_{\text{painting}}$ at level $n-1$, and equations to recall the definitions of these objects at lower levels.

5.5 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 present an intermediate "recursive definition", phrased as:

```
Fixpoint leR (n m : nat) : SProp :=
match n, m with
| O, _ => STrue
| S n, O => SFalse
| S n, S m => leR n m
end.
```

Here SProp is a definitionally proof-irrelevant impredicative universe morally[f] living at the bottom of the universe hierarchy (Gilbert et al., 2019). By placing the definition in SProp, we have definitional equality of all inequality proofs. For the purpose of unification, however, this definition does not go far enough. Consider the unification problems:

```
leR_trans ?p leR_refl = ?p
leR_trans leR_refl ?p = ?p
```

where `leR_trans` is transitivity, `leR_refl` is reflexivity, and `?p` is an existential variable. These two problems definitionally hold in SProp, but equating them does not solve the existential. For unification to be useful in inferring existentials, we present our first variant of $\leq$, which we dub as the "Yoneda variant":

```
Definition leY n m :=
  forall p, leR p n -> leR p m.
```

This definition is an improvement over `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 existentials, and we have to provide them explicitly.

The second variant of $\leq$, the "inductive variant", is phrased as:

```
Inductive leI : nat -> nat -> Type :=
| leI_refl n : n <~ n
| leI_down {n p} : p.+1 <~ n -> p <~ n
where "n <~ m" := (leI n m) : nat_scope.
```

Compared to `leY`, `leI` has no proof-irrelevance properties. This definition is specially crafted for painting, 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.

---

[f]In Coq (2023), it is however a stand-alone universe unrelated to the universe hierarchy.

5.6 Groupoid properties of equality and basic type isomorphisms

The construction relies on the groupoid properties of equality which are left implicit in table 5'. The use of the equivalence between $u = v$ and $\Sigma(p : u.\text{hd} = v.\text{hd}).(u.\text{tl} = v.\text{tl})$ for u and v in a Σ-type is left implicit in the same table. Also 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.

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

6. Future work

The construction could be extended with degeneracies as well as with permutations (Grandis and Mauri, 2003). Dependent v-sets could also be defined, opening the way to construct Π-types and Σ-types of v-sets. A v-set of v-sets representing a universe could also be defined (as sketched e.g. in a talk by Herbelin at the HoTT-UF workshop for the bridge case (2020)). More generally, we believe these lines of work to eventually provide alternative models to parametric type theories (Nuyts et al., 2017; Cavallo and Harper, 2020).

By equipping the universe construction with a structure of equivalences, as suggested along the lines of Altenkirch and Kaposi (2015), we also suspect the construction to be able to serve as a basis for syntactic models of various versions of cubical type theory (Bezem et al., 2013; Cohen et al., 2018; Angiuli et al., 2021), saving the detour via the fibered approach inherent to usual presheaf models. In particular, we wonder whether the indexed approach might provide definitional properties which are otherwise lost when detouring via presheaves. In particular, we conjecture being able to justify univalence holding definitionally. Our approach would also definitively ground cubical type theory in iterated parametricity.

Another direction for future work would also be to generically construct the indexed form of any presentation of presheaves over a direct category and to show the equivalence between the two presentations.

References

Altenkirch, T., **Capriotti, P.**, **and Kraus, N.** 2016. Extending homotopy type theory with strict equality. In **Talbot, J. and Regnier, L.**, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pp. 21:1–21:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Altenkirch, T. and Kaposi, A. 2015. Towards a cubical type theory without an interval. In **Uustalu, T.**, editor, *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia*, volume 69 of *LIPIcs*, pp. 3:1–3:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Angiuli, C., **Brunerie, G.**, **Coquand, T.**, **Harper, R.**, **(Favonia), K. H.**, **and Licata, D. R.** 2021. Syntax and models of cartesian cubical type theory. *Math. Struct. Comput. Sci.*, 31(4):424–468.

Annenkov, D., **Capriotti, P.**, **and Kraus, N.** 2017. Two-level type theory and applications. *CoRR*, abs/1705.03307.

Atkey, R., **Ghani, N.**, **and Johann, P.** 2014. A relationally parametric model of dependent type theory. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*.

Bernardy, J., **Coquand, T.**, **and Moulin, G.** 2015. A presheaf model of parametric type theory. *Electr. Notes Theor. Comput. Sci.*, 319:67–82.

Bernardy, J., **Jansson, P.**, **and Paterson, R.** 2010. Parametricity and dependent types. In **Hudak, P. and Weirich, S.**, editors, *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pp. 345–356. ACM.

Bernardy, J. and Lasson, M. 2011. Realizability and parametricity in pure type systems. In *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pp. 108–122. Springer.

Bernardy, J.-P. and Moulin, G. 2012. A computational interpretation of parametricity. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pp. 135–144. IEEE.

Bezem, M., **Coquand, T.**, **and Huber, S.** 2013. A model of type theory in cubical sets. In **Matthes, R. and Schubert, A.**, editors, *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22-26, 2013, Toulouse, France*, volume 26 of *LIPIcs*, pp. 107–128. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Buchholtz, U. and Morehouse, E. 2017. Varieties of cubical sets. In *International Conference on Relational and Algebraic Methods in Computer Science*, pp. 77–92. Springer.

Cavallo, E. and Harper, R. 2020. Internal Parametricity for Cubical Type Theory. In **Fernández, M. and Muscholl, A.**, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 13:1–13:17, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Cohen, C., **Coquand, T.**, **Huber, S.**, **and Mörtberg, A.** 2018. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In **Uustalu, T.**, editor, *TYPES 2015*, volume 69 of *LIPIcs*, pp. 5:1–5:34. Schloss Dagstuhl.

Curien, P.-L., **Garner, R.**, **and Hofmann, M.** 2014. Revisiting the categorical interpretation of dependent type theory. *Theoretical Computer Science*, 546:99–119. Models of Interaction: Essays in Honour of Glynn Winskel.

de Moura, L. M., **Kong, S.**, **Avigad, J.**, **van Doorn, F.**, **and von Raumer, J.** 2015. The lean theorem prover (system description). In **Felty, A. P. and Middeldorp, A.**, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pp. 378–388. Springer.

Gilbert, G., **Cockx, J.**, **Sozeau, M.**, **and Tabareau, N.** 2019. Definitional proof-irrelevance without K. *Proc. ACM Program. Lang.*, 3(POPL):3:1–3:28.

Grandis, M. and Mauri, L. 2003. Cubical sets and their site. *Theory and Applications of Categories*, 11:185–211.

Herbelin, H. 2015. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science*, 25(5):1116–1131.

Herbelin, H. 2020. Investigations into syntactic iterated parametricity and cubical type theory.

Hofmann, M. 1995. *Extensional concepts in intensional type theory*. Phd, University of Edinburgh.

Johann, P. and Sojakova, K. 2017. Cubical categories for higher-dimensional parametricity. *CoRR*, abs/1701.06244.

Kapulkin, K. and Lumsdaine, P. L. 2021. The simplicial model of univalent foundations (after voevodsky). *Journal of the European Mathematical Society*, 23(6):2071–2126.

Kraus, N. 2021. Internal ∞-categorical models of dependent type theory : Towards 2LTT eating HoTT. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pp. 1–14. IEEE.

Kraus, N. and Sattler, C. 2017. Space-valued diagrams, type-theoretically. *arXiv preprint arXiv:1704.04543*.

Lasson, M. 2014. *Réalisabilité et Paramétricité dans les Systèmes de Types Purs*. Ph.D. thesis, Université de Lyon.

Loregian, F. and Riehl, E. 2020. Categorical notions of fibration. *Expositiones Mathematicae*, 38(4):496–514.

Martin-Löf, P. 1975. An intuitionistic theory of types, predicative part. In *Logic Colloquium*, pp. 73–118. North Holland.

Martin-Löf, P. 1984. *Intuitionistic type theory*, volume 1 of *Studies in proof theory*. Bibliopolis.

Moeneclaey, H. 2021. Parametricity and semi-cubical types. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pp. 1–11. IEEE.

Moeneclaey, H. 2022. *Cubical models are cofreely parametric*. PhD thesis, Université Paris Cité.

Moulin, G. 2016. *Internalizing parametricity*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology.

Nuyts, A., **Vezzosi, A.**, **and Devriese, D.** 2017. Parametric quantifiers for dependent type theory. *Proc. ACM Program. Lang.*, 1(ICFP).

Part, F. and Luo, Z. 2015. Semi-simplicial types in logic-enriched homotopy type theory. *CoRR*, abs/1506.04998.

Reynolds, J. C. 1983. Types, abstraction and parametric polymorphism. In **Mason, R. E. A.**, editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*, pp. 513–523. North-Holland/IFIP.

Shulman, M. 2014. Homotopy type theory should eat itself (but so far, its too big to swallow). Blog post, homotopytypetheory.org/2014/03/03/hott-should-eat-itself.

The Agda Team 2023. *Agda User Manual, Release 2.6.4*.

The Coq Development Team 2023. *The Coq Reference Manual, version 8.18.0*.

The Univalent Foundations Program 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study.

Voevodsky, V. 2012. Semi-simplicial types. Briefly described in Section 8 of Herbelin (2015).