On the Formalization of the Simplicial Model of HoTT

Kunhong Du

Born 18.02.2000 in Guangdong, China

05.02.2025

Master's Thesis Mathematics

Advisor: Prof. Dr. Floris van Doorn

Second Advisor: Prof. Dr. Peter Koepke

Mathematisches Institut

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

1 Introduction **1**

2 Contextual categories **3**
- 2.1 The category associated to a type theory . . . 3
- 2.2 Contextual categories . . . 5
- 2.3 Logical structures on a contextual category . . . 7

3 Consistency, models and initiality **9**
- 3.1 Consistency . . . 9
- 3.2 Models and initiality . . . 10
- 3.3 The formalization of initiality . . . 10

4 Locally cartesian closed categories **11**
- 4.1 Definitions and first properties . . . 11
- 4.2 Dealing with chosen pullbacks . . . 12

5 Building contextual categories from universes **17**
- 5.1 Universes . . . 17
- 5.2 Logical structures on a universe . . . 19

6 The simplicial model **27**
- 6.1 Fibres and well-ordered morphisms . . . 27
- 6.2 Construction of the universe . . . 29
  - 6.2.1 The functor $\mathbf{W}_\alpha$ . . . 29
  - 6.2.2 The simplicial sets W_α and $\tilde{W}_\alpha$ . . . 32
  - 6.2.3 The functor $\mathbf{U}_\alpha$, the simplicial sets U_α and $\tilde{U}_\alpha$ . . . 35
- 6.3 Logical structures . . . 37

7 Conclusion **39**

A Type theory **41**
- A.1 Martin-Löf type theory . . . 41
  - A.1.1 Structural rules . . . 41
  - A.1.2 Logical rules . . . 42
  - A.1.3 The univalence axiom . . . 43
- A.2 Pure type systems . . . 44
  - A.2.1 Definition . . . 44
  - A.2.2 The syntactic category of PTS . . . 45

Chapter 1

Introduction

Homotopy Type Theory (HoTT) is a modern foundation of mathematics that integrates concepts from type theory and homotopy theory, providing a rich and versatile framework for reasoning about structures and transformations in mathematics. [6] One of the cornerstone principles of HoTT is the univalence axiom, which asserts that equivalent mathematical structures (types) can be identified. The univalence axiom has profound implications for the expressive power and philosophical interpretation of mathematics within HoTT.

A critical challenge in HoTT is to ensure the consistency of the univalence axiom. This task was first addressed by Kapulkin, Lumsdaine and Voevodsky who constructed a model in which the univalence axiom is validated. [8] Their work introduced the simplicial model, which is based on the category of Kan complexes. This construction not only confirmed the internal consistency of the univalence axiom but also provided deep insights into the interplay between type theory and homotopy theory.

To rigorously formalize the simplicial model, the use of formal proof assistants becomes essential. Lean, a powerful interactive theorem prover, has emerged as a leading tool for the formalization of mathematics. Moreover, Mathlib, a Lean library, provides rich content in category theory and simplicial sets, laying a solid foundation for our purpose. Leveraging these capabilities, this thesis aims to formalize key aspects of the simplicial model in Lean, bridging the gap between theoretical constructs and their mechanized formalizations.

By developing a formal and verifiable framework for parts of the simplicial model, this work not only seeks to deepen our comprehension of HoTT and its semantics but also highlights the practical utility of proof assistants like Lean in advancing modern mathematics.

Chapter 2 introduces contextual categories as a framework for modeling type theories, where objects are viewed as contexts and morphisms as substitutions. We also present the term model a type theory in this framework. Some details on the formalization are discussed.

In Chapter 3, we clarify the meaning of consistency of a type theory and explain how it is proven through the initiality of the term model and the existence of a model.

Chapter 4 serves as an interlude to the main construction, presenting the basic theory of locally cartesian closed categories, the categorical setting in which our constructions take place. In particular, we discuss strategies for handling chosen pullbacks in Lean

Chapter 5 reduces the construction of a contexutal categorical model to a universe and its associated logical structures, automatically solving the coherence issues in this process.

Chapter 6 presents the core theory of the simplicial model. Roughly, types are viewed as Kan fibrations in this model.

Chapter 2

Contextual categories

2.1 The category associated to a type theory

As a heuristic, we consider the following construction.

Example 2.1.1. *Let $\mathbf{T}$ be a Martin-Löf type theory (see Appendix A.1). Then its syntactic category $\mathcal{C}(\mathbf{T})$ is defined as follows:*

- *The objects are the contexts $[x_1{:}A_1, ..., x_n{:}A_n]$ up to definitional equality and renaming of free variables.*

- *A morphism*
$$f : [x_1{:}A_1, ..., x_n{:}A_n] \to [y_1{:}B_1, ..., y_m : B_m]$$
is an equivalence class of sequences of terms $(f_1, ..., f_m)$ such that
$$\begin{aligned} x_1{:}A_1, ..., x_n{:}A_n &\vdash f_1{:}B_1 \\ x_1{:}A_1, ..., x_n{:}A_n &\vdash f_2{:}B_2[f_1/y_1] \\ &\cdots \\ x_1{:}A_1, ..., x_n{:}A_n &\vdash f_m{:}B_m[f_1, ..., f_{m-1}/y_1, ..., y_{m-1}] \end{aligned}$$
where two sequences $(f_1, ..., f_m)$ and $(g_1, ..., g_m)$ are equivalent if for each i
$$x_1{:}A_1, ..., x_n{:}A_n \vdash f_i = g_i : B_i[f_1, ..., f_{i-1}/y_1, ..., y_{i-1}].$$
Note that the morphism really encodes the data we need to do substitution, so we can write $y[f]$ for the substitution $y[f_1, ..., f_n/y_1, ..., y_n]$.

- *The identity morphism of $[x_1{:}A_1, ..., x_n{:}A_n]$ is exactly $(x_1, ..., x_n)$.*

- *The composition is defined as follows. Given*
$$[x_1{:}A_1, ..., x_n{:}A_n] \xrightarrow{(f_1,...,f_m)} [y_1{:}B_1, ..., y_m{:}B_m] \xrightarrow{(g_1,...,g_k)} [z_1{:}C_1, ..., z_k{:}C_k],$$
$(g_1, ..., g_k) \circ (f_1, ..., f_m)$ is
$$(g_1[f], ..., g_k[f]),$$
i.e., the sequence after substitution.

$\mathcal{C}(\mathbf{T})$ has a tree structure.

1. the root is the empty context [].
2. the depth (grading) is the length of the context.
3. the root [] is the only object with zero depth.
4. any object except for the root has a unique parent. For all $n \in \mathbb{N}$, the parent of $[x_1{:}A_1, ..., x_{n+1}{:}A_{n+1}]$ is $[x_1{:}A_1, ..., x_n{:}A_n]$.
5. There is a canonical map from the child to its parent:
$$[x_1{:}A_1, ..., x_{n+1}{:}A_{n+1}] \xrightarrow{(x_1,...,x_n)} [x_1{:}A_1, ..., x_n{:}A_n],$$
which is called the canonical projection and denoted by p.

Judgments viewed in $\mathcal{C}(\mathbf{T})$

Derivable judgments in $\mathbf{T}$ are one-to-one corresponded to morphisms (if we identify objects with the identity morphisms) in $\mathcal{C}(\mathbf{T})$.

$\Gamma \vdash A$ $\mathsf{type} \Leftrightarrow \Gamma, x{:}A$ $\mathsf{ctx} \Leftrightarrow$ an object in $\mathcal{C}(\mathbf{T})$.

$\Gamma \vdash a{:}A \Leftrightarrow$ A section of the canonical projection $p : [\Gamma, x{:}A] \to [\Gamma]$: If $\Gamma \vdash a{:}A$ is derivable, then so is $\Gamma \vdash A$ type. There is a morphism $(\mathrm{id}_\Gamma, a) : [\Gamma, x{:}A] \to [\Gamma]$ such that $p \circ (\mathrm{id}_\Gamma, a) = (\mathrm{id}_\Gamma) \circ (\mathrm{id}_\Gamma, a) = \mathrm{id}_{[\Gamma, x:A]}$. Conversely, if there is a section $s : [\Gamma, x{:}A] \to [\Gamma]$, then by definition of morphisms in $\mathcal{C}(\mathbf{T})$, the last term of s is a derivable judgment $\Gamma \vdash a : A$ for some a.

Judgmental equalities in $\mathbf{T}$ clearly correspond to the strict equality in $\mathcal{C}(\mathbf{T})$.

Remark. It follows from our discussion that $\mathcal{C}(\mathbf{T})$ contains exactly all the information of $\mathbf{T}$. With $\mathcal{C}(\mathbf{T})$, we can basically put aside the syntax and work with type theory categorically.

Substitutions in $\mathcal{C}(\mathbf{T})$

Let $\Gamma = [x_1{:}A_1, ..., x_n{:}A_n, x_{n+1}{:}A_{n+1}]$ and $\Delta = [y_1{:}B_1, ..., y_m{:}B_m]$. Let
$$f = (f_1, ..., f_m) : \Delta \to \mathrm{ft}\,\Gamma = (x_1{:}A_1, ..., x_n{:}A_n).$$
Define
$$f^*\Gamma := [y_1{:}B_1, ..., y_m{:}B_m, y_{m+1}{:}A_{n+1}[f]]$$
and
$$q(f) := (f_1, ..., f_n, y_{m+1}).$$

Note that $\mathrm{ft}(f^*\Gamma) = \Delta$, hence we have the following diagram

$$\begin{tikzcd}
{[y_1{:}B_1, ..., y_m{:}B_m, y_{m+1}{:}A_{n+1}[f]]} \arrow[r, "q(f)"] \arrow[d, "p_{f^*\Gamma}"'] & {[x_1{:}A_1, ..., x_{n+1}{:}A_{n+1}]} \arrow[d, "p_\Gamma"] \\
{[y_1{:}B_1, ..., y_m{:}B_m]} \arrow[r, "f"'] & {[x_1{:}A_1, ..., x_n{:}A_n]}
\end{tikzcd}$$

It clearly commutes. Furthermore, it is a pullback: given compatible $(g_1, ..., g_m) : \Delta' \to \Delta$ and $(h_1, ..., h_{n+1}) : \Delta' \to \Gamma$, the unique lift is $(g_1, ..., g_m, h_{n+1}) : \Delta \to f^*\Gamma$.

This construction is exactly the counterpart of (simultaneous) syntactic substitution: given a context $[x_1{:}A_1, ..., x_n{:}A_n]$, one of its extension $[x_1{:}A_1, ..., x_{n+1}{:}A_{n+1}]$ and a sequence of terms $(f_1, ..., f_n)$ under a context Δ, we can substitute all the x_i's in A_{n+1} and obtain an extension of Δ. Note that by the universal property of pullbacks, we can also do substitution of terms in $\mathcal{C}(\mathbf{T})$.

On feature that needs noting is that the (chosen) pullbacks are strictly functorial. We check it commutes with composition. With the data in the diagram above, we consider a morphism

$$g = (g_1, ..., g_m) : \Delta' = [z_1{:}C_1, ..., z_k{:}C_k] \to \Delta.$$

Then

$$A_{n+1}[f_1, ..., f_n/x_1, ..., x_n][g_1, ..., g_m/y_1, ..., y_m] = A_{n+1}[g \circ f/x_1, ..., x_n]$$

by design. Hence $(f \circ g)^*\Delta = f^*(g^*\Delta)$ and

$$q(f \circ g) = (f \circ g, z_{k+1}) = (f, y_{m+1}) \circ (g, z_{k+1}) = q(f) \circ q(g).$$

Other structural rules like Var, Wkg are also validated automatically in $\mathcal{C}(\mathbf{T})$.

2.2 Contextual categories

The notion of contextual categories dates back to [4]. The key point is to package the structural rules of a type theory into a category.

Definition 2.2.1 [`ContextualCategory`]. *A contextual category is a category $\mathcal{C}$ with the following data:*

1. *a grading* $\mathrm{gr} : \mathrm{Ob}\,\mathcal{C} \to \mathbb{N}$. *Denote* Ob_n *for objects with grading* n.
2. *an object* 1 *such that*
   - *(a) 1 is the unique object with grading 0.*
   - *(b) 1 is terminal is* $\mathcal{C}$.
3. *for each* $n \in \mathbb{N}$*, a map* $\mathrm{ft}_n : \mathrm{Ob}_{n+1}\,\mathcal{C} \to \mathrm{Ob}_n\,\mathcal{C}$. *Since no confusion will arise, we will suppress the subscripts.*
4. *for each* $n \in \mathbb{N}$ *and* $X \in \mathrm{Ob}_{n+1}\,\mathcal{C}$*, a map (called the canonical projection)* $p_X : X \to \mathrm{ft}(X)$.
5. *for each* $n \in \mathbb{N}$*,* $X \in \mathrm{Ob}_{n+1}\,\mathcal{C}$ *and* $f : Y \to \mathrm{ft}\,X$*, an object* f^*X *with a map* $q(f) : f^*X \to X$ *such that*
   - *(a)* $\mathrm{ft}(f^*X) = X$.
   - *(b) the diagram*

$$\begin{tikzcd} f^*X \arrow[r, "q(f)"] \arrow[d, "p_{f^*X}"'] & X \arrow[d, "p_X"] \\ Y \arrow[r, "f"] & \mathrm{ft}\,X \end{tikzcd}$$

is a pullback (called the canonical pullback).

(c) *these pullbacks are strictly functorial (in other words, q is functorial): for all $X \in \mathrm{Ob}_{n+1}(X)$,*

i. $1^*_{\mathrm{ft}\,X} X = X$ *and* $q(1_{\mathrm{ft}\,X}) = 1_X$.

ii. *for* $f : Y \to \mathrm{ft}\,X$ *and* $g : Z \to Y$, $(f \circ g)^* X = g^*(f^* X)$ *and* $q(f \circ g) = q(f) \circ q(g)$.

$\mathcal{C}(\mathbf{T})$ is obviously a contextual category. It also tells us how to view judgments of a type theory in a contextual category:

Context	Γ	Object	$[\![\Gamma]\!]$
Type statement	$\Gamma \vdash A$ type	Canonical projection	$[\![p_{\Gamma.A}]\!]$
Typing statement	$\Gamma \vdash x{:}A$	Section of the projection	
Judgmental equality		(Strict) equality	

Before addressing the formalization, we should make a few comments on the general principles of formalizing category theory in Lean.

In Mathlib, the type of morphisms of a category is defined dependently as follows:

```
class Quiver (V : Type u) where
  Hom : V → V → Sort v
```

Therefore, morphisms with only propositionally equal sources and targets do not have the same type, which results in typing errors when we compare them with `Eq` or other dependent types. To avoid such issues, it is best to require that the equalities between objects are definitional. Or at least, it would be better to ensure the morphisms we want to compare have definitionally equal sources and targets. When this is not feasible—for instance, when the equalities of objects are additional data—use `eqToHom` or `HEq` instead of `rw` or `cast`, as the latter approach usually leads to the so-called dependent type theory hell.

```
class PreContextualCategory (α : Type u) extends Category α where
   gr : α → ℕ
   one : α
   one_gr : gr one = 0
   one_uniq {x : α} : gr x = 0 → x = one
   one_terminal : IsTerminal one
   ft : α → α
   ft_one : ft one = one
   ft_gr {n : ℕ} (x : α): gr x = n + 1 → gr (ft x) = n
   proj (x : α) : x ⟶ ft x

class PreContextualCategory.NR {α : Type u} [PreContextualCategory α]
      (x : α) : Prop where
   nr : gr x ≠ 0

class ContextualCategory (α : Type u) extends PreContextualCategory α where
  pb {x y : α} [NR x] (f : y ⟶ ft x) : α
  pb_fst {x y : α} [NR x] (f : y ⟶ ft x) : pb f ⟶ x
  gr_pb {x y : α} [NR x] {f : y ⟶ ft x} : gr (pb f) ≠ 0
  nr_pb {x y : α} [NR x] {f : y ⟶ ft x} : NR (pb f) := ⟨gr_pb⟩
  ft_pb {x y : α} [NR x] {f : y ⟶ ft x} : ft (pb f) = y
  isPullback {x y : α} [NR x] (f : y ⟶ ft x) :
    IsPullback (pb_fst f) (proj (pb f) ≫ eqToHom ft_pb) (proj x) f
  pullback_id_obj {x : α} [NR x]: pb (𝟙 (ft x)) = x
  pullback_id_map {x : α} [NR x] :
    HEq (pb_fst (𝟙 (ft x))) (𝟙 x)
  pullback_comp_obj {x y z : α} [NR x] {f : y ⟶ ft x} {g : z ⟶ y} :
    pb (g ≫ f) = pb (g ≫ eqToHom (ft_pb (f := f)).symm)
  pullback_comp_map {x y z : α} [NR x] {f : y ⟶ ft x} {g : z ⟶ y} :
    HEq (pb_fst (g ≫ f)) (pb_fst (g ≫ eqToHom ft_pb.symm) ≫ pb_fst f)
```

The first consideration is that, since the chosen pullbacks are only defined for non-root objects, it would be easiest to define a class for them to avoid repeatedly writing out the explicit proof. Therefore, we split the definition into two parts.

Another point to mention is the use of `eqToHom` and `HEq`. As discussed above, since `ft (pb f)` and `y` are only equal propositionally as extra data, it is unavoidable to connect, for example, morphisms to `ft(pb f)` and morphisms from `f` via `eqToHom` to pass the type-check. We choose `HEq` over `eqToHom ... ≫ ... = ...` due to its clarity in presentation. They are essentially equivalent:

```
lemma eqToHom_comp_iff_heq {X X' Y : C} {f : X ⟶ Y} {g : X' ⟶ Y}
   (hX : X' = X) :
     eqToHom hX ≫ f = g ↔ HEq f g
```

2.3 Logical structures on a contextual category

If $\mathbf{T}$ has further logical rules, $\mathcal{C}(\mathbf{T})$ will be equipped with the corresponding data tautologically. For example, if $\mathbf{T}$ has Π-types, then $\mathcal{C}(\mathbf{T})$ validates the corresponding rules. This will be clear by considering the formation rule of Π-types. Suppose we have an object Γ and the extension $\Gamma.A.B$. Then by definition of $\mathcal{C}(\mathbf{T})$, Γ is a (well-formed) context and we have $\Gamma, x : A \vdash B$ type. By Π-Form, we have $\Gamma \vdash \Pi_{x:A}B$, which in turn gives an object $\Gamma.\Pi_{x:A}B$ as an extension of Γ. Similarly, every other rule holds tautologically.

This inspires us to define extra structures on a contextual category to serve as the categorical counterparts of logical rules. Before that, let us denote $(A_0, A_1, ..., A_n)$ for an object in $\mathcal{C}$ such that

$$\mathrm{ft}^i(A_0, A_1, ..., A_n) = (A_0, A_1, ..., A_{n-i})$$

for all $1 \leqslant i \leqslant n$. In particular, $(A_0, A_1, ..., A_n)$ has length $\geqslant n$. We also denote p_{A_i} to be the canonical projection $(A_0, ..., A_i) \to (A_0, ..., A_{i-1})$. When we quantify over $(A_0, A_1, ..., A_n)$, we really mean to quantify over an object A_0 and its extensions of length n.

Definition 2.3.1 [`ContextualCategory.Pi_type`]. *A Π-type structure on a contextual category $\mathcal{C}$ consists of:*

1. *(Π-Form) for all (Γ, A, B), an object $(\Gamma, \Pi(A, B))$.*

2. *(*Π*-Intro) for all* (Γ, A, B) *and a section* $b : (\Gamma, A) \to (\Gamma, A, B)$*, a section* $\lambda(b) : \Gamma \to (\Gamma, \Pi(A, B))$.

3. *(*Π*-Elim) for all* (Γ, A, B) *and sections* $k : \Gamma \to (\Gamma, \Pi(A, B))$ *and* $a : \Gamma \to (\Gamma, A)$*, a map* $\mathsf{app}(k, a) : a \to p_B$ *in the slice category of over* Γ.

4. *(*Π*-Comp) for all* (Γ, A, B) *and sections* $a : \Gamma \to (\Gamma, A)$ *and* $b : (\Gamma, A) \to (\Gamma, A, B)$*, we have* $\mathsf{app}(\lambda(b), a) = b \circ a$.

5. *(Stable under substitution) for all* $f : \Delta \to \Gamma$*, and suitable* $(\Gamma, A, B), a, b, k$*, we have*

$$\begin{aligned} f^*(\Pi(A, B)) &= \Pi(f^*A, f^*B) \\ \lambda(f^*b) &= f^*\lambda(b) \\ \mathsf{app}(f^*k, f^*a) &= f^*(\mathsf{app}(k, a)) \end{aligned}$$

Remark. The third and forth term are a little different than the direct translation of syntax. For Π-Elim, the conclusion $\Gamma \vdash f(a) : B(a)$ should correspond to the section of $(\Gamma, B[a/x])$. But it is essentially equivalent by the universal property of pullback. Unfolding this equivalence, term 4 is equivalent as well.

Similarly we can define other logical structures on a contextual category $\mathcal{C}$ (even for type theory other than MLTT, see Appendix A.2.2). As we discussed above, $\mathcal{C}(\mathbf{T})$ is equipped with all the type structures corresponded to those logical rules of $\mathbf{T}$ in a completely tautological way.

Chapter 3

Consistency, models and initiality

3.1 Consistency

As mentioned in the introduction, the consistency of the univalence axiom, or the consistency of HoTT remained unresolved until the discovery of the simplicial model. But what exactly do we mean by consistency?

Definition 3.1.1. *A type theory is inconsistent if all types are inhabited. It is consistent if it is not inconsistent.*

In particular, it is clear from the elimination rule of the empty type $\mathsf{0}$ that $\mathbf{T}$ with $\mathsf{0}$ is inconsistent if and only if $\mathsf{0}$ is inhabited by a closed term (i.e, a term with type $\mathsf{0}$ under empty context).

Theorem 3.1.2. *MLTT is consistent.*

Section A.4 of [6] presents a sketch of the proof. The idea is that: every term in MLTT can be normalized to a normal form and there is no term in normal form having type 0.

However, after adding the univalence axiom, the previous argument no long applies, since the term $\mathsf{univalence}(A, B)$ does not simplify.

Remark. The reader may notice when talking about the system of type theory, we make a difference between rules and axioms. This is not because of the form of presentation. In fact, all the rules and axioms can be presented in either proof tree or in if-then clauses. The distinction lies in the computational nature. Rules in the form of "formation, introduction and elimination" are computational, justified by the corresponding computation rule, while the computational interpretation of axioms is not guaranteed. In particular, the univalence has no computational interpretation when formulated as an axiom in the form of Definition A.1.1.

To prove the consistency of HoTT, we may want to continue the work on MLTT. For the latest progress in this approach, see [2], where HoTT is proved to have homotopy canonicity. Alternatively, we may also approach the problem by exhibiting a semantic model.

3.2 Models and initiality

Having mentioned the word “model” several times, it’s time to clarify what we mean by it and what we want with it.

Definition 3.2.1. *A model of type theory in contextual categories is a contextual category with all the logical structures corresponding to the logical rules and axioms of the type theory.*

Remark. We emphasis that what we define are “models in contextual categories” because type theory can be modelled in other structures, such as categories with attributes (which are very similar to contextual categories), categories with families, display map categories, etc.

In contrast to the classical model theory settings, having a model does not imply the consistency of a type theory. In fact, for any type theory $\mathbf{T}$, $\mathcal{C}(\mathbf{T})$ is a model of $\mathbf{T}$, albeit it contains only tautological information and hence says nothing about consistency. Therefore, to argue about the consistency, we need the following initiality theorem.

Theorem 3.2.2. *If $\mathcal{C}$ is a model of $\mathbf{T}$, then there is a unique functor from $\mathcal{C}(\mathbf{T})$ to $\mathcal{C}$ preserving all the contextual structures. In other words, $\mathcal{C}(\mathbf{T})$ is initial among all the models of $\mathbf{T}$.*

Suppose $\mathbf{T}$ is inconsistent, i.e., $\mathsf{0}$ is inhabited in $\mathcal{C}(\mathbf{T})$. By the initiality theorem, the interpretation of $\mathsf{0}$ is inhabited in every model of $\mathbf{T}$. Therefore, to prove the consistency of $\mathbf{T}$, it suffices to find a model where the interpretation of $\mathsf{0}$ is not inhabited.

To conclude, the ultimate goal is to construct a contextual category (theoretically in ZFC or practically in Lean) with all the logical structures of MLTT and the univalence axiom such that the interpretation of $\mathsf{0}$ is not inhabited. In doing so, the consistency of HoTT (relative to ZFC or Lean) is proved in light of initiality.

3.3 The formalization of initiality

Historically, the correctness of Theorem 3.2.2 was a subject of debate, as no one had provided a complete proof for sufficiently large type theories, particularly MLTT, until [10]. In [8], it is still regarded as a conjecture.

The initiality of MLTT with Π, Σ, Id and a hierarchy of universes was formalized in Agda [3]. Although we would hope to achieve the same in Lean, the task is too extensive to complete within the given time constraints. However we have succeeded in formalizing a relatively smaller type theory, called pure type systems, and also its syntactic category (see Appendix A.2). With this, we argue that it is possible to formalize MLTT, its syntactic category and initiality in Lean, leaving it for further research.

Chapter 4

Locally cartesian closed categories

Before delving into the construction, we present here the basic theory of locally cartesian closed categories, which serves as the categorical settings for our work. The reasons for this choice will be explained below

4.1 Definitions and first properties

Definition 4.1.1. *A category $\mathcal{C}$ with finite products is cartesian closed if for any object Y, the functor $- \times Y : \mathcal{C} \Rightarrow \mathcal{C}$ has a right adjoint $[Y, -]$.*

It is well known that there is an adjunction

$$\{\text{theories in simply typed } \lambda\text{-calculus}\} \leftrightarrows \{\text{cartesian closed categories}\}.$$

When moving from simple type theory to dependent type theory, one need a slight generalization of Definition 4.1.1.

Definition 4.1.2. *Let $\mathcal{C}$ be a category with pullbacks. It is said to be locally cartesian closed if one the following equivalent properties holds:*

1. *(`CategoryTheory.LocallyCartesianClosed`). for any morphism $f : X \to Y$, the pullback functor $f^* : \mathcal{C}/Y \to \mathcal{C}/X$ has a right adjoint $\Pi_f : \mathcal{C}/X \to \mathcal{C}/Y$.*
2. *(`CategoryTheory.OverCartesianClosed`). for any object X, the slice category $\mathcal{C}/X$ is cartesian closed.*

Remark. In a category with pullbacks, any choice of pullbacks along a morphism f form a functor. In Lean, it is chosen by by `Classical.choose`. See Section 4.2 for detailed discussion.

Remark. Π_f provides an interpretation of the formation of Π-types. In fact, (extensional) Martin-Löf type theories are biequivalent to locally cartesian closed categories [5]. This is reason why it is chosen as the setting we work in.

The second property is the reason why it's named "locally" cartesian closed.

Only the proof that the second property implies the first one has been formalized [`CategoryTheory.LocallyCartesianClosed.instOfOverCartesianClosed`], since we don't need the other direction. The formalization is entirely attributed to [7].

We will see in Chapter 6 that the simplicial model heavily relies on the fact that simplicial sets form a locally cartesian closed category. In fact, it is true for all presheaf categories.

Theorem 4.1.3 [`CategoryTheory.instLocallyCartesianClosedPresheaves`]. *Any presheaf category is locally cartesian closed.*

The proof has the two steps:

1. (`CategoryTheory.instCartesianClosedPresheaves`) Any presheaf category is cartesian closed.

2. (`CategoryTheory.CategoryOfElements.OverIsoElementsPresheaves`) For a presheaf P in a presheaf category $\mathcal{C}^{op} \Rrightarrow \mathrm{Set}$, the slice category over P is equivalent to the presheaf category $P^{el} \Rrightarrow \mathrm{Set}$. (See the remark below Theorem 6.2.8).

The proof strategy is inspired by [7]. Both of these statements follow directly from theorems already in Mathlib. We omit the details here. But we would like to make one comment on the universe issues.

Generally, we define a presheaf category to have codomain as the universe that the morphisms lie in. In Lean, it means for `C : Type u` with an instance `Category.{v,u} C`, the presheaf category is `C`^{op} $\Rrightarrow$ `Type v`. Indeed, almost all the results related to presheaves in Mathlib are formalized in this way.

However, `SSet`, which is viewed as the presheaf category on the simplex category, is defined polymorphically in Lean : `SSet.{u} := simplexCategory` $\Rrightarrow$ `Type u`, where the objects and morphisms of `simplexCategory` are in `Type 0`. As a result, those results on presheaves in Mathlib do not apply to `SSet`. In order to use them here, we need to slightly modify the codes. In particular, we need to replace every occurrence of

1. `C`^{op} $\Rrightarrow$ `Type v` by `C`^{op} $\Rrightarrow$ `Type max v w`. Then it applies to `SSet` since in Lean `Type (max 0 u) = Type u` definitionally.

2. `yoneda` by `yonedaULift`, a polymorphic version of `yoneda`. Roughly, it's `yoneda` postcomposed with `uliftFunctor`

3. `yonedaEquiv` by `yonedaULiftEquiv`.

Though the modified versions are indeed the generalization conceptually, but they are not corollaries of each other in Lean, since `ULift.{0,0}` is not definitionally equal to `id`. Their relation is more comparable to lemmas of groups written in '$+$' or '$\cdot$'.

4.2 Dealing with chosen pullbacks

When formalizing categories, general issue arises: we cannot treat isomorphic objects as identical, as is often done in informal mathematics. The same thing happens when we want to apply the adjunction $f^* \dashv \Pi_f$ to the pullbacks given by a universe (Definition 5.1.1), they simply do not match.

The first solution that might come to mind is to modify the definition of locally cartesian closed categories by including chosen pullbacks as additional data and defining the adjunctions based on them. However, this approach may complicate the proofs of related properties. Furthermore, a universe provides only *some* pullbacks, which are insufficient to define a pullback functor for every morphism in the category. Define some pullbacks via the universe and the others using `Classical.choice`? No, this is definitely the last thing we want to do in Lean.

It turns out that perhaps the best solution is to formalize the informality, i.e., identify isomorphic objects via isomorphisms. Let $f : X \to Y$ be a morphism and $f^* \dashv \Pi_f$ an adjunction. Let $W \to X$ be a random pullback of $Z \to Y$ along f, as indicated in the following diagram. We have a unique isomorphism from $W \to f^*Z$ in $\mathcal{C}/X$. Then for any $P \to X$, we have an bijection

$$\mathrm{Hom}_{\mathcal{C}/X}(W, P) \cong \mathrm{Hom}_{\mathcal{C}/Y}(Z, \Pi_f P)$$

given by composing the isomorphism with $\mathrm{Hom}_{\mathcal{C}/X}(f^*Z, P) \cong \mathrm{hom}_{\mathcal{C}/Y}(Z, \Pi_f P)$.

$$\begin{tikzcd}
 & f^*Z \arrow[r] \arrow[drr] \arrow[dd, dashed] & P \arrow[ddl, dashed] & & \Pi_f P \arrow[ddl] \\
W \arrow[ur, "\cong"] \arrow[urr] \arrow[rrr] \arrow[dr] \arrow[dr, phantom, "\lrcorner", very near start] & & & Z \arrow[ur] \arrow[d] & \\
 & X \arrow[rr, "f"] & & Y &
\end{tikzcd}$$

The bijection has an obvious naturality on P. For $h : P \to P'$,

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{C}/X}(W, P) \arrow[r] \arrow[d, "h\circ-"] & \mathrm{Hom}_{\mathcal{C}/Y}(Z, \Pi_f P) \arrow[d, "\Pi_f h\circ-"] \\
\mathrm{Hom}_{\mathcal{C}/X}(W, P') \arrow[r] & \mathrm{Hom}_{\mathcal{C}/Y}(Z, \Pi_f P')
\end{tikzcd}$$

The naturality on the left is slightly trickier. For $h : Z \to Z'$ and two pullback squares as in the following diagram, there is a lift $l : W \to W'$ by the universal property of pullbacks.

$$\begin{tikzcd}
 & W' \arrow[r] \arrow[rrr, bend left] \arrow[dd, dashed] & P \arrow[ddl, dashed] & & Z' \arrow[r] \arrow[dd, dashed] & \Pi_f P \arrow[ddl] \\
W \arrow[ur, "l"] \arrow[urr] \arrow[rrr] \arrow[dr] \arrow[dr, phantom, "\lrcorner", very near start] & & & Z \arrow[ur, "h"] \arrow[urr] \arrow[dr] & & \\
 & X \arrow[rrr, "f"] & & & Y &
\end{tikzcd}$$

The following square commutes

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{C}/X}(W', P) \arrow[r] \arrow[d, "-\circ l"] & \mathrm{Hom}_{\mathcal{C}/Y}(Z', \Pi_f P) \arrow[d, "-\circ h"] \\
\mathrm{Hom}_{\mathcal{C}/X}(W, P) \arrow[r] & \mathrm{Hom}_{\mathcal{C}/Y}(Z, \Pi_f P)
\end{tikzcd}$$

The bijections together with the naturality on both sides provide all the information of the adjunction. They work smoothly with the a random pullback, in particular, the chosen pullbacks of a universe.

Now, we can formalize the following lemmas. Note that we use $-^{tr}$ to denote transpose along the adjunction in either direction.

Lemma 4.2.1.

$$\begin{tikzcd}
 & & & \Pi_g A \arrow[d] \arrow[dr] & \\
A \arrow[d, "f"] \arrow[dr, "i"] & & & C \arrow[dr, "k"] & \Pi_{g'} A' \arrow[d] \\
B \arrow[rrru, dashed, "g"] \arrow[dr, "j"] & A' \arrow[d, "f'"] & & & C' \\
 & B' \arrow[rrru, "g'"] & & &
\end{tikzcd}$$

Given a diagram as above, if the bottom square is a pullback then there is natural map $\Pi_g A \to \Pi_{g'} A'$ *making the right square commute.* [`CategoryTheory.LocallyCartesianClosed.pushforward.isPullbackLift`] [`CategoryTheory.LocallyCartesianClosed.pushforward.isPullbackLift_fst`]

If the left square is also a pullback, then so is the right square. [`CategoryTheory.LocallyCartesianClosed.pushforward.isPullback`]

Proof. For $f : X \to Y$, we denote the functor $\mathcal{C}/X \to \mathcal{C}/Y : g \mapsto f \circ g$ by Σ_f.

The map $\Pi_g A \to \Pi_{g'} A'$ is given as follows:

First notice that for any φ in $\mathcal{C}/C$, there is a unique isomorphism $g'^* \circ \Sigma_k \varphi \cong \Sigma_j \circ g^* \varphi$. Then consider the map

$$g'^* \Sigma_k \Pi_g f \cong \Sigma_j g^* \Pi_g f \xrightarrow{\Sigma_j \varepsilon_g(f')} \Sigma_j f \xrightarrow{i} f',$$

where ε_g is the counit.

Taking the transpose via the adjunction $g'^* \dashv \Pi_{g'}$ gives us the desired map.

The commutativity is because the transpose is a map $\Sigma_k \Pi_g f \to \Pi_{g'} f'$ in the $\mathcal{C}/C'$.

Now assume the left square is also a pullback. We give a informal proof that the right one is also a pullback, meaning that we identify all the isomorphic objects as identical. This is of course not true and should be taken care of when doing formalization. More concretely, we need to use the alternative versions of adjunction we present above.

$$\begin{tikzcd}
 & & D \arrow[dr, "w'"] \arrow[ddrr, bend left, "v"] \arrow[ddr, bend right, "u"'] & & \\
g^*D \arrow[dr, "w"] \arrow[ddrr, bend left, "v'"] \arrow[ddr, bend right, "g^*u"'] & & & \Pi_g A \arrow[d] \arrow[dr, "\phi"] & \\
 & A \arrow[d, "f"] \arrow[dr, "i"] & & C \arrow[dr, "k"] & \Pi_{g'}A' \arrow[d] \\
 & B \arrow[urr, dashed, "g"] \arrow[dr, "j"] & A' \arrow[d, "f'"] & & C' \\
 & & B' \arrow[urr, "g'"] & &
\end{tikzcd}$$

Let $u : D \to C$ and $v : D \to \Pi_{g'}A'$ be maps such that the outer square commutes. We want to prove there exists a unique w' such that the two triangle commutes.

For existence, let v' be the transpose of v under $g'^* \dashv \Pi g'$. Notice that we identify g^*D with g'^*D. Then on the left, the condition of lift is satisfied, as

$$j \circ g^*u = g'^*(k \circ u) = g'^*(\Pi_g f' \circ v) = f' \circ v'$$

where the first two equalities are unstrict identification of pullbacks, the last one follows from the naturality of adjunction.

Now there is a lift $w : g^*D \to A$, which give the desired w' via the adjunction:
1. $\Pi_g f \circ w' = u$ by design.
2. $\phi \circ w' = (i \circ \varepsilon)^{tr} \circ w' = ((i \circ \varepsilon) \circ g^*w')^{tr} = (i \circ w)^{tr} = v'^{tr} = v$. The second equality follows from naturality.

Uniqueness simply follows from the fact that, via the adjunction, every lift on the right corresponds to lift on the left, which is unique. □

Lemma 4.2.2 [`CategoryTheory.LocallyCartesianClosed.pushforward.adj_lift_eq_lift_adj`]**.** *In the settings of the Lemma 4.2.1, suppose further there are pullbacks $E - D - B - C$ and $E' - D' - B' - C'$, a map $d : D \to D'$, and a commutative square $D - D' - C - C'$. Let $e : E \to E'$ denote the lift.*

$$\begin{tikzcd}
E \arrow[r, "\widehat{b^{tr}}"] \arrow[dr, "e"] \arrow[dd] & A \arrow[dr, "i"] \arrow[ddl, dashed] & & D \arrow[r, "\hat{b}"] \arrow[dr, "d"] \arrow[dd] & \Pi_g A \arrow[dr, "\varphi"] \arrow[ddl, dashed] & \\
 & E' \arrow[r, "b^{tr}"] \arrow[dd] & A' \arrow[ddl] & & D' \arrow[r, "b"] \arrow[dd] & \Pi_{g'}A' \arrow[ddl] \\
B \arrow[rrr, dashed, "g"] \arrow[dr] & & & C \arrow[dr] & & \\
 & B' \arrow[rrr] & & & C' &
\end{tikzcd}$$

Let $b : D' \to \Pi_{g'}A'$. Denote $b^{tr} : E' \to A'$ the transpose and $\hat{b} : E \to A$ the lift. Then the lift $\widehat{b^{tr}} : D \to \Pi_g A$ coincides with the transpose $\hat{b}^{tr}$.

Also, let $a : E' \to A'$. We have $\widehat{a^{tr}} = \hat{a}^{tr}$.

Proof. By the uniqueness of lift, it suffices to show that the following squares commutes

$$\begin{tikzcd}
E \arrow[r, "\hat{b}^{tr}"] \arrow[d, "e"'] & A \arrow[d, "i"] \\
E' \arrow[r, "b^{tr}"'] & A'
\end{tikzcd}$$

We compute,

$$\begin{aligned}
i \circ \hat{b}^{tr} &= i \circ \varepsilon \circ g^*(\hat{b}) \\
&= (\varphi \circ \hat{b})^{tr} \\
&= (b \circ d)^{tr} \\
&= b^{tr} \circ e
\end{aligned}$$

It follows that $\widehat{b^{tr}} = \hat{b}^{tr}$. Applying this to a^{tr} obtains $\widehat{a^{tr}} = \hat{a}^{tr}$. □

Lemma 4.2.3 [`CategoryTheory.LocallyCartesianClosed.pushforward.trans_comp`]**.** *The construction in Lemma 4.2.1 is functorial, i.e., in the following diagram, with two pullback squares on the bottom and two commutative squares on the left, the map* $\Phi : \Pi_g A \to \Pi_{g''} A''$ *given by the pasted squares is the composition of the maps* φ, φ' *obtained by small squares.*

$$\begin{tikzcd}
 & & \Pi_g A \arrow[d] \arrow[dr, "\varphi"] \arrow[ddrr, bend left, "\Phi"] & & \\
A \arrow[d, "f"] \arrow[dr, "i"] & & C \arrow[dr, "k"] & \Pi_{g'} A' \arrow[d] \arrow[dr, "\varphi'"] & \\
B \arrow[urr, dashed, "g"] \arrow[dr, "j"] & A' \arrow[d, "f'"] \arrow[dr, "i'"] & & C' \arrow[dr, "k'"] & \Pi_{g''} A'' \arrow[d] \\
 & B' \arrow[urr, dashed, "g'"] \arrow[dr, "j'"] & A'' \arrow[d] & & C'' \\
 & & B'' \arrow[urr, "g''"] & &
\end{tikzcd}$$

Proof. Denote $\varepsilon : g^*\Pi_g A \to A$, $\varepsilon' : g'^*\Pi_{g'} A' \to A'$ to be the counits. Then

$$\begin{aligned}
\Phi &= (i' \circ i \circ \varepsilon)^{tr} \\
&= (i' \circ \varepsilon' \circ g'^* \varphi)^{tr} \\
&= (i' \circ \varepsilon')^{tr} \circ \varphi \\
&= \varphi' \circ \varphi
\end{aligned}$$

□

Chapter 5

Building contextual categories from universes

5.1 Universes

The major obstruction to constructing a contextual categories lies in choosing suitable pullbacks such that they are strictly functorial. To solve this, we introduce the notion of universe. This "universe" does correspond to the type-theoretic universe, but here the naming mainly refers to its universality in a categorical sense.

Definition 5.1.1 [`Universe`]. *Let $\mathcal{C}$ be a category. A universe in U is a pair of objects U, U' and a morphism $p : \tilde{U} \to U$, and for each map $f : X \to U$, an object $(X; f)$ and a pair of morphisms $P_f : (X; f) \to f$ and $Q_f : (X; f) \to \tilde{U}$ such that the square*

$$\begin{tikzcd}
(X;f) \arrow[r, "Q_f"] \arrow[d, "P_f"'] & \tilde{U} \arrow[d, "p"] \\
X \arrow[r, "f"] & U
\end{tikzcd}$$

is a pullback.

Definition 5.1.2 [`Universe.Chain`]. *Fix a universe $p : \tilde{U} \to U$. We define the notion of $(X; f_1, ..., f_n)$ inductively.*

1. *When $n = 0$, $(X;) = X$.*
2. *If $(X; f_1, ..., f_{n-1})$ is defined and $f_n : (X; f_1, ..., f_{n-1}) \to U$, then $(X; f_1, ..., f_n)$ is the chosen pullback of f_n along p.*

The motivation behind the definition is that we want to define a contextual category where every canonical projection is a pullback along p. As the following diagram implies, in that case, $q(g)$ can be chosen as the lift of pullback. By the property of the pullbacks, the left square is again a pullback.

$$\begin{tikzcd}
(Y; f \circ g) \arrow[r, dashed, "q(g)"'] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] \arrow[rr, bend left] & (X; f) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d] \\
Y \arrow[r, "g"] & X \arrow[r, "f"] & U
\end{tikzcd}$$

To formalize Definition 5.1.2 in Lean, we use inductive families.

```
inductive Chain (U : Universe C) : C → C → Type (max u v)
| nil X : U.Chain X X
| cons (Y : C) (g : Y ⟶ U.down) (c : U.Chain X Y)  : U.Chain X (U.pt g)
```

This definition has the following good properties

1. Every `Chain` is either `nil` or `cons Y g c`. This property allows us to do convenient case analysis when proving propositions of `Chain`.

2. `ft (cons Y g c) = Y` definitionally, consistent with our principles of formalizing categories in Lean.

The following more direct approach, i.e., lists of morphisms satisfying certain propositions, does not enjoy these properties. That's why we prefer the inductive definitions over it.

```
structure Chain (U : Universe C) (X : C) where
  obj : List (Σ x : C, x ⟶ U.obj)
  well_formed : Formed X obj
```

Indeed, we can construct a contextual category out of any universe in a category with a terminal object.

Definition 5.1.3 [`Universe.Chains.instChainsContextualCategory`]**.** *Given a category $\mathcal{C}$ with a universe U and a terminal object 1, we define $\mathcal{C}_U$ as follows:*

- $\mathrm{Ob}\,\mathcal{C}_U := \{(f_1, ..., f_n) | f_i : (1; f_1, ..., f_{i-1}) \to U$ *for all* $1 \leqslant i \leqslant n\}$.

- $\mathcal{C}_U((f_1, ..., f_n), (g_1, ..., g_m)) := \mathcal{C}((1; f_1, ..., f_n), (1; g_1, ..., g_m))$ *with identity morphisms and compositions in* $\mathcal{C}$.

- $\mathrm{gr} :=$ *length of the sequence*

- $1_{C_U} := ()$*, the empty sequence.*

- $\mathrm{ft}(f_1, ..., f_{n+1}) = \mathrm{ft}(f_1, ..., f_n)$.

- $p_{(f_1,...,f_{n+1})} := P_{f_{n+1}}$

- *Given* $(f_1, .., f_{n+1})$ *and* $\alpha : (g_1, ..., g_m) \to (f_1, ..., f_n)$.

$$\alpha^*(f_1, ..., f_n) := (g_1, ..., g_m, f_{n+1} \circ \alpha),$$

and $q(\alpha)$ *is given by the lift the pullback as in the following diagram:*

$$\begin{tikzcd}
(1; g_1, ..., g_m, f_{n+1} \circ \alpha) \arrow[dd, "p=P_{f_{n+1}\circ\alpha}"'] \arrow[dr, dashed, "q(\alpha)"] \arrow[drr, "Q_{f_{n+1}\circ\alpha}"] \arrow[ddr, phantom, "\lrcorner"] & & \\
& (1; f_1, ..., f_{n+1}) \arrow[r, "Q_{f_{n+1}}"'] \arrow[d] \arrow[dr, phantom, "\lrcorner"] & \tilde{U} \arrow[d, "p"] \\
(1; g_1, ..., g_m) \arrow[r, "\alpha"] & (1; f_1, ..., f_n) \arrow[r, "f_{n+1}"] & U
\end{tikzcd}$$

Proposition 5.1.4 [`Universe.Chains.instChainsContextualCategory`]. *$\mathcal{C}_U$ is a contextual category.*

Proof. The properties about pullbacks are the only nontrivial ones to check.

First note that $Q(f_{n+1} \circ \alpha) = Q_{f_{n+1}} \circ q(\alpha)$, and hence the fact that the big outer square and the right square are pullbacks in $\mathcal{C}$ imply that the left square is a pullback $\mathcal{C}$. Also, a square in $\mathcal{C}_U$ is a pullback if it is a pullback when viewed in $\mathcal{C}$. Hence $q(\alpha)$ together with $P_{f_{n+1}\circ\alpha}$ forms a pullback in $\mathcal{C}_U$.

Clearly $\mathrm{id}^*(f_1, ..., f_n) = (f_1, ..., f_n)$ and it follows from the uniqueness of the lift that $q(\mathrm{id})$ must also be identity.

Now let $\beta : (h_1, ..., h_k) \to (g_1, ..., g_m)$. By the associativity of composition, $\alpha^*\beta^* = (\beta \circ \alpha)^*$. Again, $q(\alpha \circ \beta) = q(\alpha) \circ q(\beta)$ follows from the uniqueness of the lift.

$$\begin{tikzcd}
(1; h_1, ..., h_k, f_{n+1} \circ \alpha \circ \beta) \arrow[dd] \arrow[dr, "q(\beta)"] \arrow[drr, "q(\alpha\circ\beta)"] \arrow[drrr] & & & \\
& (1; g_1, ..., g_m, f_{n+1} \circ \alpha) \arrow[r, "q(\alpha)"'] \arrow[d] & (1; f_1, ..., f_{n+1}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] \\
(1; h_1, ..., h_k) \arrow[r, "\beta"] & (1; g_1, ..., g_m) \arrow[r, "\alpha"] & (1; f_1, ..., f_n) \arrow[r, "f_{n+1}"] & U
\end{tikzcd}$$

□

The formalization of the proof is straightforward, with only one trick to note. To prove any properties on non-nil chains (`NR c` in the codes), it suffices to prove them for `cons Y p c`, since any non-nil chain has this form.

```
def cases_cons {h : (c : U.Chains t) → [NR c] → Sort w}
   (h' : ∀ {Y p c}, h (cons Y p c)) (c : U.Chains t) [NR c] :
      h c
```

The observation particularly applies to those properties about pullbacks in the definition of contextual categories. In this case, `ft_pb` holds definitionally:

```
lemma ft_pb_cons: ft (pb_cons f) = d := rfl
```

This would help us eliminate some occurrences of `eqToHom` and slightly simplify the arguments.

5.2 Logical structures on a universe

From the previous section, we know that given any universe U, we can construct a contextual category $\mathcal{C}_U$. The next question is: what data of U is required to equip $\mathcal{C}_U$ with various logical structures? We will focus on Π-type in this paper.

Let $\mathcal{C}$ be locally cartesian closed (Definition 4.1.2).

(For the formalization of Definition 5.2.1 and Proposition 5.2.2, see the file `TypeStructures.lean`)

Definition 5.2.1. *Given a universe U in $\mathcal{C}$ in a locally category closed category $\mathcal{C}$, define $\pi_1 : U \times \tilde{U} \to U$ and $\pi_2 : U \times \tilde{U} \to \tilde{U}$ to be projections. Define U^{Π} to be the domain of $\Pi_p \pi_2$, which is illustrated as*

$$\begin{tikzcd}
U \times \tilde{U} \arrow[d, "\pi_2"'] & U^{\Pi} \arrow[d, "\Pi_p \pi_2"] \\
\tilde{U} \arrow[r, "p"'] & U
\end{tikzcd}$$

Pulling back $\Pi_p \pi_2$ along p induces $\alpha_{\mathsf{gen}} : A_{\mathsf{gen}} \to U^{\Pi}$. The counit of the adjunction gives a morphism

$$\alpha''_{\mathsf{gen}} : A_{\mathsf{gen}} = p^*(U \times \tilde{U}) \to U \times \tilde{U}$$

or more precisely $p^* \Pi_p(\pi_2) \to \pi_2$, as illustrated as the dashed line:

$$\begin{tikzcd}
A_{\mathsf{gen}} \arrow[r, dashed] \arrow[dr, "p^*\Pi_p\pi_2"'] & U \times \tilde{U} \arrow[d, "\pi_2"] & & U^{\Pi} \arrow[r, "\mathrm{id}"] \arrow[dr, "\Pi_p\pi_2"'] & U^{\Pi} \arrow[d, "\Pi_p\pi_2"] \\
& \tilde{U} \arrow[rrr, "p"'] & & & U
\end{tikzcd}$$

Define $\alpha'_{\mathsf{gen}} : A_{\mathsf{gen}} \to U$ to be $\pi_1 \circ \alpha''_{\mathsf{gen}}$. Pulling back α'_{gen} along p induces $\beta_{\mathsf{gen}} : B_{\mathsf{gen}} \to A_{\mathsf{gen}}$.

To conclude, we have a diagram

$$\begin{tikzcd}
B_{\mathsf{gen}} \arrow[rr] \arrow[d, "\beta_{\mathsf{gen}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \tilde{U} \arrow[d] \\
A_{\mathsf{gen}} \arrow[r] \arrow[rr, bend left, "\alpha'_{\mathsf{gen}}"] \arrow[d, "\alpha_{\mathsf{gen}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] & U \\
U^{\Pi} \arrow[r, "\Pi_p\pi_2"'] & U &
\end{tikzcd}$$

The subscript gen stands for "generic" as we have the following property:

Proposition 5.2.2. *Let $B \overset{p_2}{\to} A \overset{p_1}{\to} \Gamma$ be a sequence with maps $\delta : \Gamma \to U$, $\gamma : A \to \tilde{U}$, $\delta' : A \to U$ and $\gamma' : B \to \tilde{U}$ exhibiting $A \to \Gamma$ as pullbacks of p:*

$$\begin{tikzcd}
A \arrow[r, "\gamma"] \arrow[d, "p_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] & B \arrow[r, "\gamma'"] \arrow[d, "p_2"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] \\
\Gamma \arrow[r, "\delta"'] & U & A \arrow[r, "\delta'"'] & U
\end{tikzcd}$$

Then there is a unique morphism $\Gamma \to U^{\Pi}$ we denote by $\ulcorner A, B \urcorner$ such that we have a diagram

$$\begin{tikzcd}
B \arrow[r] \arrow[d, "p_2"'] \arrow[dr, phantom, "\lrcorner", very near start] & B_{gen} \arrow[rr] \arrow[d, "\beta_{gen}"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \tilde{U} \arrow[d] \\
A \arrow[r] \arrow[d, "p_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & A_{gen} \arrow[r] \arrow[d, "\alpha_{gen}"'] \arrow[rr, bend left, "\alpha'_{gen}"] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] & U \\
\Gamma \arrow[r, "\ulcorner A,B \urcorner"'] & U^{\Pi} \arrow[r, "\Pi_p \pi_2"'] & U &
\end{tikzcd}$$

where $\Pi_p\pi_2 \circ \ulcorner A, B \urcorner = \delta$ and similar equations hold for the other maps. In other words, $B \overset{p_2}{\to} A \overset{p_1}{\to} \Gamma$, δ, γ, δ' and γ' are exhibited by pullbacks and compositions.

Proof. The proof is divided into 5 steps.

1. Note that γ is the pullback of δ along p. Define $\ulcorner A, B \urcorner$ to be given by $\langle \delta', \gamma \rangle : A \to U \times \tilde{U}$ under the adjunction:

$$\begin{tikzcd}
A \arrow[r, "\langle \delta', \gamma \rangle"] \arrow[dr, "\gamma"'] & U \times \tilde{U} \arrow[d, dashed, "\pi_2"] & & \Gamma \arrow[r, dashed, "\ulcorner A,B \urcorner"] \arrow[dr, "\delta"'] & U^{\Pi} \arrow[d, "\Pi_p \pi_2"] \\
& \tilde{U} \arrow[rrr, "p"'] & & & U
\end{tikzcd}$$

In particular, $\delta = \Pi_p\pi_2 \circ \ulcorner A, B \urcorner$.

2. Define $l : A \to A_{\mathsf{gen}}$ to be the lift.

$$\begin{tikzcd}
A \arrow[dr, dashed, "l"] \arrow[drr, "\gamma"] \arrow[dd, "p_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \\
& A_{\mathsf{gen}} \arrow[r] \arrow[d, "\alpha_{\mathsf{gen}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] \\
\Gamma \arrow[r, "\ulcorner A,B \urcorner"'] & U^{\Pi} \arrow[r, "\Pi_p \pi_2"'] & U
\end{tikzcd}$$

Note that the right square is a pullback as desired. Or, if we flip the square,

$$\begin{tikzcd}
A \arrow[r, dashed, "l"] \arrow[d, "\gamma"'] \arrow[drr, "f"] & A_{\mathsf{gen}} \arrow[dl] \arrow[drr, "\alpha_{\mathsf{gen}}"] & & \\
\tilde{U} \arrow[drr, "p"] & & \Gamma \arrow[r, "\ulcorner A,B \urcorner"] \arrow[d, "\delta"'] & U^{\Pi} \arrow[dl, "\Pi_p \pi_2"] \\
& & U &
\end{tikzcd}$$

l is the pullback of $\ulcorner A, B \urcorner$ along p.

3. We need to examine $\delta' = \alpha'_{\mathsf{gen}} \circ l$. This follows from the naturality of adjunction. Recall $l = p^* \ulcorner A, B \urcorner$ and α''_{gen} is the counit:

$$\begin{tikzcd}
A \arrow[r, "l"] \arrow[dr] & A_{\mathsf{gen}} \arrow[r, "{\alpha_{\mathsf{gen}}}''"] \arrow[d] & U \times \tilde{U} \arrow[dl] & \Gamma \arrow[r, "\ulcorner A,B \urcorner"] \arrow[dr] & U^{\Pi} \arrow[r, "\mathrm{id}"] \arrow[d] & U^{\Pi} \arrow[dl] \\
& \tilde{U} \arrow[rrr, "p"] & & & U &
\end{tikzcd}$$

Hence we have

$$\begin{tikzcd}
\mathrm{id} \arrow[r] \arrow[d, mapsto, "-\circ\ulcorner A,B\urcorner"'] & \alpha''_{\mathsf{gen}} \arrow[d, mapsto, "-\circ l"] \\
\ulcorner A,B\urcorner \arrow[r, mapsto] \arrow[dr, mapsto] & \alpha''_{\mathsf{gen}}\circ l \arrow[d, equal] \\
& \langle\delta',\gamma\rangle
\end{tikzcd}$$

It follows that $\delta' = \pi_1\circ\langle\delta',\gamma\rangle = \pi_1\circ\alpha''_{\mathsf{gen}}\circ l = \alpha'_{\mathsf{gen}}\circ l$

4. $B\to B_{\mathsf{gen}}$ is defined to be the lift as in step 2.

5. To prove the uniqueness of $\ulcorner A,B\urcorner$, it suffices to show it corresponds to $\langle\delta',\gamma\rangle$ under the adjunction. Let φ be such a map. Using the same argument as in step 3, it is reduced to show that $\alpha''_{\mathsf{gen}}\circ p^*\varphi = \langle\delta',\gamma\rangle$.

For the sake of clarity, we denote the pullback of $\Pi_p\pi_2$ along p by q.

$$\begin{tikzcd}
A \arrow[dr, dashed, "p^*\varphi"] \arrow[drr, "\gamma"] \arrow[dd, "p_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \\
& A_{\mathsf{gen}} \arrow[r, "q"] \arrow[d, "\alpha_{\mathsf{gen}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d, "p"] \\
\Gamma \arrow[r, "\varphi"'] & U^{\Pi} \arrow[r, "\Pi_p\pi_2"'] & U
\end{tikzcd}$$

Then $q\circ p^*\varphi = \gamma$ since it's lift, which implies $\pi_2\circ\alpha''_{\mathsf{gen}}\circ p^*\varphi = \gamma$. Also, $\alpha'\circ p^*\varphi = \delta'$ by assumption, which implies $\pi_1\circ\alpha''_{\mathsf{gen}}\circ p^*\varphi = \delta'$. It follows that $\alpha''_{\mathsf{gen}}\circ p^*\varphi = \langle\delta',\gamma\rangle$. □

The formalization is straightforward, with heavy use of adjusted bijections described in Section 4.2.

Definition 5.2.3 [`Universe.Pi.Structure`]. *A* Π*-structure on a universe* U *consists of a map*

$$\Pi : U^{\Pi}\to U$$

with an isomorphism, over U^{Π}

$$\Pi^*\tilde{U}\cong \Pi_{\alpha_{gen}}B_{gen}.$$

It is equivalent to data of a map $\tilde{\Pi}$ *such that the square is a pullback:*

$$\begin{tikzcd}
B_{gen} \arrow[d, "\beta_{gen}"'] & \Pi_{\alpha_{gen}}B_{gen} \arrow[r, "\tilde{\Pi}"] \arrow[d, "\Pi_{\alpha_{gen}}\beta_{gen}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d] \\
A_{gen} \arrow[r, "\alpha_{gen}"] & U^{\Pi} \arrow[r, "\Pi"] & U
\end{tikzcd}$$

Theorem 5.2.4 [`Universe.Chains.Pi_type`]. *A* Π*-structure on a universe induces a* Π*-type structure on* $\mathcal{C}_U$*.*

Proof. Recall Π-type structure is defined in Definition 2.3.1.

1. (Π-Form) Recall in $\mathcal{C}_U$, every (Γ, A, B) corresponds to pullbacks along certain maps.

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "f"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d] & & B \arrow[r] \arrow[d, "g"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U} \arrow[d] \\
\Gamma \arrow[r, "\ulcorner A \urcorner"] & U & & A \arrow[r, "\ulcorner B \urcorner"] & U
\end{tikzcd}$$

Define $\Pi(A, B)$ to be the canonical pullback of $\Pi \circ \ulcorner A, B \urcorner : \Gamma \to U$.

2. (Π-Intro) Given a section $b : (\Gamma, A) \to (\Gamma, A, B)$, we want to construct a section $\lambda(b) : \Gamma \to (\Gamma, \Pi(A, B))$.

a) First, we obtain a map b^{tr} via adjunction

$$\begin{tikzcd}
A \arrow[r, "b"] \arrow[dr, "\mathrm{id}"'] & B \arrow[d, "g"'] & & \Gamma \arrow[r, dashed, "b^{tr}"] \arrow[dr] & \Pi_f B \arrow[d, "\Pi_f g"] \\
& A \arrow[rrr, "f"'] & & & \Gamma
\end{tikzcd}$$

b) Recall by Proposition 5.2.2, we have the follow diagram:

$$\begin{tikzcd}
& & \Pi_f B \arrow[d] \arrow[dr, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & \\
B \arrow[d, "g"'] \arrow[dr] \arrow[dr, phantom, "\lrcorner", very near start] & & \Gamma \arrow[dr, "\ulcorner A, B \urcorner"] & \Pi_{\alpha_{\mathsf{gen}}} B_{\mathsf{gen}} \arrow[d] \\
A \arrow[urr, "f"] \arrow[dr] & B_{\mathsf{gen}} \arrow[d, "\beta_{\mathsf{gen}}"] & & U^{\Pi} \\
& A_{\mathsf{gen}} \arrow[urr, "\alpha_{\mathsf{gen}}"] & &
\end{tikzcd}$$

By Lemma 4.2.1, there exists a map $\Pi_f B \to \Pi_{\alpha_{\mathsf{gen}}} B_{\mathsf{gen}}$ such that the right square is a pullback. Therefore, $\Pi_f B \cong \ulcorner A, B \urcorner^* \Pi_{\alpha_{\mathsf{gen}}} B_{\mathsf{gen}}$. Here the pullback is given by canonical pullback of $\Pi^* \tilde{U}$ composed with the isomorphism in Definition 5.2.3.

c) In the slice category over Γ, we have

$$\Pi_f g \cong \ulcorner A, B \urcorner^* \Pi_{\alpha_{\mathsf{gen}}} \beta_{\mathsf{gen}} = \ulcorner A, B \urcorner^* \Pi^* p = (\Pi \circ \ulcorner A, B \urcorner)^* p.$$

The last two equalities follow from our choice of the pullbacks. Finally, b^{tr} composed with this isomorphism gives a section of $(\Pi \circ \ulcorner A, B \urcorner)^* p$.

$$\begin{tikzcd}
\Gamma \arrow[r, dashed, "b^{tr}"] \arrow[dr] & \Pi_f B \arrow[r, "\cong"] \arrow[d, "\Pi_f g"] & (\Pi \circ \ulcorner A, B \urcorner)^* B \arrow[dl, "(\Pi \circ \ulcorner A, B \urcorner)^* p"] \\
& \Gamma &
\end{tikzcd}$$

3. (Π-Elim) Given sections $h : \Gamma \to (\Gamma, \Pi(A, B))$ and $a : \Gamma \to (\Gamma, A)$, we want a section $\mathsf{app}(k, a) : \Gamma \to (\Gamma, A, B)$.

   Again we compose the isomorphism $\Pi(A, B) := (\Pi \circ \ulcorner A, B \urcorner)^* \tilde{U} \cong \Pi_f B$ with f and obtain a section of $\Pi_f B$. We still denote it by f by abusing the notation. Then consider:

$$\begin{tikzcd}
\ast \arrow[r, "\epsilon"] \arrow[dr] & B \arrow[d] & \Pi_f B \arrow[r, "\mathrm{id}"] \arrow[d] & \Pi_f B \arrow[dl] \\
& A \arrow[r, "f"] \arrow[ul, dashed, bend left, "l"] & \Gamma \arrow[l, bend left, "a"] \arrow[u, bend left, "h"] &
\end{tikzcd}$$

   where $*$ is the pullback of $\Pi_f B$, the dashed line l is the section induced by pullback (as a lift) and ϵ is the counit of the adjunction. Then $\epsilon \circ l \circ a$ is the required section.

4. (Π-Comp) Given sections $a : \Gamma \to (\Gamma, A)$ and $b : (\Gamma, A) \to (\Gamma, A, B)$, we have $\mathsf{app}(\lambda(b), a) = b \circ a$.

   Unfolding the isomorphisms $\Pi(A, B) \cong \Pi_{A \to \Gamma} B$ we have used for every construction, we obtain

$$\mathsf{app}(\lambda(b), a) := \epsilon \circ l \circ a,$$

   where ϵ is the counit of $\Pi_f B$ as in (Π-elim) and l is the lift of b^{tr} as in (Π-intro).

   Then it suffices to show $\epsilon \circ l = b$, which follows from naturality.

$$\begin{tikzcd}
A \arrow[r, "l"] \arrow[rr, bend left, "b"] \arrow[dr] & \ast \arrow[r, dashed, "\epsilon"] \arrow[d] & B \arrow[dl] & \Gamma \arrow[r, "b^{tr}"] \arrow[dr] & \Pi_{A \to \Gamma} B \arrow[r, "\mathrm{id}"] \arrow[d] & \Pi_{A \to \Gamma} B \arrow[dl] \\
& A \arrow[rrr] & & & \Gamma &
\end{tikzcd}$$

5. (Stable under substitution) for all $f : \Delta \to \Gamma$, and suitable $(\Gamma, A, B), a, b, k$, we have

$$f^*(\Pi(A, B)) = \Pi(f^*A, f^*B) \tag{1}$$
$$\lambda(f^*b) = f^*\lambda(b) \tag{2}$$
$$\mathsf{app}(f^*k, f^*a) = f^*(\mathsf{app}(k, a)) \tag{3}$$

   (1) Note that $\ulcorner f^*A, f^*B \urcorner = \ulcorner A, B \urcorner \circ f$ by uniqueness. Hence

$$\begin{aligned}
\Pi(f^*A, f^*B) &= (\Pi \circ (\ulcorner f^*A, f^*B \urcorner))^* \tilde{U} \\
&= (\Pi \circ \ulcorner A, B \urcorner \circ f)^* \tilde{U} \\
&= f^*((\Pi \circ \ulcorner A, B \urcorner)^* \tilde{U}) \\
&= f^* \Pi(A, B)
\end{aligned}$$

(2) By Lemma 4.2.1, we have the following diagram, where i_Γ and i_Δ are the isomorphisms described in the construction of Π-Elim. Note that the upper square on the right commutes.

$$\begin{tikzcd}
 & & \Pi_{f^*\varphi}B \arrow[r, "{i_\Delta, \cong}"] \arrow[d] \arrow[rd] & \Pi(f^*A, f^*B) \arrow[ld, dashed] \arrow[rd] & \\
f^*B \arrow[d] \arrow[rd] & & \Delta \arrow[u, bend left, "(f^*b)^{tr}"] \arrow[rd, "f"] & \Pi_\varphi B \arrow[r, "{i_\Gamma, \cong}"] \arrow[d] & \Pi(A,B) \arrow[ld] \\
f^*A \arrow[u, bend left, "f^*b"] \arrow[rru, "f^*\varphi"] \arrow[rd] & B \arrow[d] \arrow[l, phantom, "\lrcorner"] & & \Gamma \arrow[u, bend right, "b^{tr}"'] & \\
 & A \arrow[u, bend right, "b"'] \arrow[rru, "\varphi"] & & &
\end{tikzcd}$$

Then

$$\begin{aligned}
\lambda(f^*b) &= i_\Delta \circ (f^*b)^{tr} \\
&= i_\Delta \circ f^*(b^{tr}) && \text{(by Lemma 4.2.2)} \\
&= f^*(i_\Gamma \circ b^{tr}) && \text{(by uniqueness of lift)} \\
&= f^*(\lambda(b))
\end{aligned}$$

(3) Note the all the squares in the following diagram commute.

$$\begin{tikzcd}
(f^*\varphi)^*\Pi_{f^*\varphi}B \arrow[r, "\epsilon_\Delta"] \arrow[d] \arrow[rrd, dashed, bend left=0] & f^*B \arrow[rd] \arrow[rdd, dashed] & & & \Pi_{f^*\varphi}B \arrow[dd] \arrow[rd] & \\
\varphi^*\Pi_\varphi B \arrow[rr, "\epsilon_\Gamma"] \arrow[rrrdd] & & B \arrow[rdd] & & & \Pi_\varphi B \arrow[dd] \\
 & & f^*A \arrow[rr, "f^*\varphi"] \arrow[rd] \arrow[rd, phantom, "\lrcorner" near start] & & \Delta \arrow[uu, bend left, "f^*k"] \arrow[ll, bend left, "f^*a"] \arrow[rd, "f"] & \\
 & & & A \arrow[rr, "\varphi"] \arrow[llluu, bend left, "\varphi^*k"] & & \Gamma \arrow[uu, bend left, "k"] \arrow[ll, bend left, "a"]
\end{tikzcd}$$

By the universal property, it suffices to show $\mathsf{app}(f^*k, f^*a) = \epsilon_\Delta \circ (f^*\varphi)^*(f^*k) \circ f^*a$ is also a lift of $\mathsf{app}(k, a)$ along f:

$$\begin{tikzcd}
\Delta \arrow[rrd, bend left, "\mathsf{app}(k,a)\circ f"] \arrow[rd, dashed, "\mathsf{app}(f^*k,f^*a)"] \arrow[rddd, bend right, "\mathrm{id}"'] & & \\
& f^*B \arrow[r, "b"] \arrow[d] & B \arrow[d, "a"] \\
& f^*A \arrow[r] \arrow[d] & A \arrow[d] \\
& \Delta \arrow[r, "f"] & \Gamma
\end{tikzcd}$$

The left square commutes because app is designed to be a section.

We can compute

$$b \circ \epsilon_\Delta \circ (f^*\varphi)^*(f^*k) \circ f^*a = \epsilon_\Gamma \circ \varphi^*k \circ a \circ f$$

by unfolding the commutativity of the squares above. Then it follows that the upper square also commutes.

$\square$

Chapter 6

The simplicial model

In this chapter, we present the core theory of the simplicial model.

6.1 Fibres and well-ordered morphisms

Definition 6.1.1 [`SSet.Fibre`]. *Let $f : X \to Y$ be a map of simplicial sets. Let $y \in Y_n$, define the fibre of y to be the preimage of y along $f_n : X_n \to Y_n$. We denote it by $f^{-1}(y)$*

Note that the terminology is not canonical, i.e., it does not refer to the categorical fibres.

Definition 6.1.2 [`SSet.WellOrderedHom`]. *A well-ordered morphism of simplicial sets consists of*

1. *A map of simplicial sets $f : X \to Y$.*
2. *For all n, and all $y \in Y_n$, a well-order on the fibre $f^{-1}(y)$.*

Remark. Recall a well-order is a well-founded linear order. It enjoys a power uniqueness property: given preorders on sets α and β, if the order on β is a well-order, then there exists at most one order isomorphism between them.

Definition 6.1.3 [`SSet.OrderIso`]. *For two well-ordered morphisms $f : X \to Y$ and $f' : X' \to Y$, an isomorphism between f and f' is an isomorphism of simplicial sets $\varphi : X \to X'$, such that*

1. *the diagram commutes :*
$$\begin{tikzcd} X \arrow[d, "\varphi{,}\cong"'] \arrow[dr, "f"] & \\ X' \arrow[r, "f'"'] & Y \end{tikzcd}$$
2. *for every $y \in Y_n$, $\varphi_n|_{f^{-1}(y)} : f^{-1}(y) \to f'^{-1}(y)$ is an order isomorphism.*

Remark. By virtue of the uniqueness property of well-orders, we observe that there exists at most one isomorphism between two well-ordered morphisms. This is crucial, as we will see later, and it is the reason why we consider well-ordered morphisms.

Now we might want to consider the collection of isomorphism classes of the well-ordered morphisms. However, simply doing so will obtain a proper class rather than a set. Therefore, we need a bound for restriction.

Definition 6.1.4 [`SSet.SmallWO`]. *Let α be a cardinal. A well-ordered morphisms $f : X \to Y$ is α-small if for every y, the cardinality of $f^{-1}(y)$ is strictly smaller than α.*

Lemma 6.1.5 [`SSet.instSmallΩ_obj₀`]. *The collection of isomorphism classes of α-small well-ordered morphisms is a set.*

Proof. We provide a sketch of proof in the informal language of set theory.

The collection of all the sets with cardinality $< \alpha$ modulo bijections

$$\{S||S| < \alpha\}/\sim$$

is a set. So is the collection

$$A = \{S_n, n \in \mathbb{N}|\forall n \in \mathbb{N}, |S_n| < \alpha\}/\sim$$

where $\sim$ is induced by the componentwise bijections. All the possible simplicial structures plus well-ordered structures on a representative of a fixed element in A is a set. Hence such union over A is a set. It is clear that the union surjects onto the collection of isomorphism classes of α-small well-ordered morphisms. It follows that the latter is a set. □

The formalization of this lemma in Lean is in a completely different style.

```
instance Setoid_SmallWO {α : Cardinal.{u}} {Y : SSet.{u}} :
      Setoid (SmallWO α Y) where -- suppose we are in ‘Type u’
  r := SmallWO.rel -- the equivalence relation as we described above
  iseqv := SmallWO.relIseqv

def Ω_obj₀ (α : Cardinal.{u}) (Y : SSet) : Type (u + 1) :=
   Quotient (@Setoid_SmallWO Y α) -- take the quotient
```

The claim of Lemma 6.1.5 should then be formalized as

```
instance : Small.{u, u + 1} (Ω_obj₀ α X)
```

Here, `Small.{w, v}` is a predicate for types in `Type v` saying that it is equivalent to a type in `Type w`

```
class Small.{w, v} (α : Type v) : Prop where
  equiv_small : ∃ S : Type w, Nonempty (α ≃ S)
```

With an instance of smallness, we can apply the function

```
def Shrink (α : Type v) [Small.{w} α] : Type w
```

allowing us to extract a type in the smaller universe. We can therefore define the key type in the construction of the model

```
def Ω_obj (α : Cardinal.{u}) (Y : SSet) : Type u :=
   Shrink (Ω_obj₀ α Y)
```

The Lean proof follows the idea we present above, the crucial construction is

```
structure SmallFibresWithStructures.{u} (α : Cardinal.{u}) (X : SSet) :
    Type u where
  fibre {n : SimplexCategoryᵒᵖ} (x : X.obj n) : Shrink (Set.Iio α)
  ... -- fields of simplicial and small fibre structures
```

Here, `Set.Iio` α is the set of all the cardinals $< \alpha$, which is initially in `Type (u + 1)` but dragged down to `Type u` by `Shrink`. Therefore, the structure itself lies in `Type u`.

Then we construct a function

```
def SmallFibresWithStructures.to:
  SmallFibresWithStructures α X → Ω_obj₀ α X
```

and prove it is surjective. Finally, by virtue of the following theorem, we prove the desired result.

```
theorem small_of_surjective {α : Type v} {β : Type w} [Small.{u} α]
  {f : α → β} (hf : Function.Surjective f) : Small.{u} β
```

6.2 Construction of the universe

6.2.1 The functor $\mathbf{W}_\alpha$

With Lemma 6.1.5, we can define

Definition 6.2.1 [`SSet.Ω`]. *The functor $\mathbf{W}_\alpha : \text{sSet}^{op} \to \text{Set}$ is defined as follows.*

1. *$\mathbf{W}_\alpha(X)$ is the set of the isomorphism classes of small well-ordered morphisms on X.*

2. *For a map of simplicial sets $\varphi : X \to Y$, and $[f : X' \to Y] \in \mathbf{W}_\alpha(Y)$, $\varphi[f] = [\varphi^* f]$ is the pullback of f along φ, with well-orders given by the pullback.*

Remark. We elaborate a little further on the details of equipping orders on the pullback. First note that, for a pullback of sets,

$$\begin{tikzcd} D \arrow[r, "i"] \arrow[d, "g"'] \arrow[dr, phantom, "\lrcorner", very near start] & C \arrow[d, "f"] \\ A \arrow[r, "h"'] & B \end{tikzcd}$$

for all $a \in A$, $i|_{g^{-1}(a)} : g^{-1}(a) \to f^{-1}(h(a))$ is a bijection. Also, for a pullback of simplicial sets, the evaluation on each layer is still a pullback. Hence, we can equip the fibers of the pullback with (well-)orders via the bijections.

Remark. For Definition 6.2.1 to be well-defined, there are several things to be checked before.

1. Taking pullback is sound with respect to the equivalence classes.

2. W_α satisfies the axioms of functor, i.e., $W_\alpha(\text{id}) = \text{id}$ and $W_\alpha(f \circ g) = W_\alpha(g) \circ W_\alpha(f)$.

The proofs are straightforward but the formalization can be tricky, because we are dealing with setoids and `Shrink` at the same time. We omit the details here.

$\mathbf{W}_\alpha$ is denoted by Ω in the formalization, and `Ω_obj` and `Ω_map` are functions from `SSet` to `Type u` rather than `SSet`^{op} to `Type u` for convenience.

Theorem 6.2.2 [`SSet.Ω.PreservesLimitsOfSize`]**.** $\mathbf{W}_\alpha$ *preserves limits, i.e., given a small diagram* $F : J \Rightarrow \mathrm{sSet}^{op}$ *with a limit cone* c*,* $(\mathbf{W}_\alpha) \circ c$ *is also a limit cone. In particular, there is an isomorphism* $\mathbf{W}_\alpha(\lim_J F) \cong \lim_J (W_\alpha \circ F)$*.*

The "small" diagrams are formalized as diagrams whose domain category has an object type lying in a smaller universe. We do not need any assumption on its morphism type.

```
instance Ω.PreservesLimitsOfSize.{u, v, w}
   (α : Cardinal.{u}) [UnivLE.{v, u}] :
      Limits.PreservesLimitsOfSize.{w, v} (Ω α)
```

Here, `Limits.PreservesLimitsOfSize.{w, v}` means the functor preservers all the limit of the functor from a category with object types in `Type v` and morphism type in `Type w`. `UnivLE` conceptually means `v` is smaller than or equal to `u`.

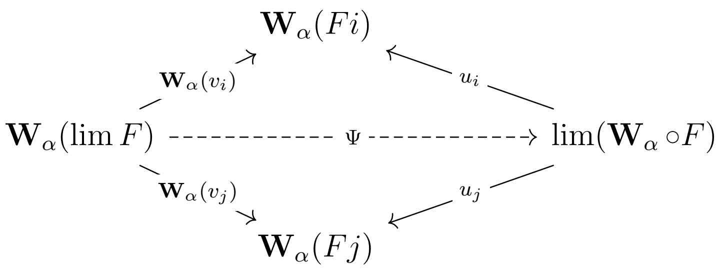
Now we present the proof along with its formalization.

Fix a small diagram $F : J \to \mathrm{sSet}^{op}$ with a limit. We have a canonical morphism from $\Psi : \mathbf{W}_\alpha(\lim F) \to \lim(\mathbf{W}_\alpha \circ F)$. The proof is finished once Ψ is proven to be an isomorphism. Since we are in the category of sets, it suffices to prove Ψ is a bijection.

Let $v_j : Fj \to \lim F, j \in J$, $u_j : \lim(\mathbf{W}_\alpha \circ F) \to \mathbf{W}_\alpha(Fj)$ be the legs of the respective limit cone.

Injectivity:

Let $f, g \in \mathbf{W}_\alpha(\lim F)$ such that $\Psi(f) = \Psi(g)$. Then their pullbacks along v_j are isomorphic for all $j \in J$, as illustrated below, where the triangles commute by design.



$$\begin{tikzcd}
 & \mathbf{W}_\alpha(Fi) & \\
\mathbf{W}_\alpha(\lim F) \arrow[ur, "\mathbf{W}_\alpha(v_i)"] \arrow[dr, "\mathbf{W}_\alpha(v_j)"'] \arrow[rr, dashed, "\Psi"] & & \lim(\mathbf{W}_\alpha \circ F) \arrow[ul, "u_i"'] \arrow[dl, "u_j"] \\
 & \mathbf{W}_\alpha(Fj) &
\end{tikzcd}$$

The rough idea is that, since fibres are invariant under pullbacks, the isomorphisms between pullbacks provide an isomorphism between f and g.

In details, the construction is divided into several steps.

First we provide a way to construct isomorphisms between well-ordered morphisms through `Pieces`. That is, we need only compatible order isomorphisms between corresponding fibres.

```
def move {f : X ⟶o Y} (φ : n ⟶ m) {y : Y.obj n} (x : f⁻¹ y) :
   f⁻¹ (Y.map φ y) := ⟨X.map φ x, ...⟩

structure Pieces where
  orderIso {n : SimplexCategoryᵒᵖ} (y : Y.obj n) : f⁻¹ y ≃o f'⁻¹ y
```

```
compatible {n m : SimplexCategory^op} (φ : n ⟶ m)
  {y : Y.obj n} (x : f⁻¹ y) :
    orderIso (Y.map φ y) (move φ x) = move φ (orderIso y x)
```

Now the problem is reduced to finding a bunch of compatible order isomorphisms $f^{-1}(x) \cong g^{-1}(x)$ with $x \in (\lim F)_n$.

Recall that a diagram of sets $\Phi : I \Rightarrow \mathrm{Set}$ with colimit has the jointly surjective property, i.e., $\sqcup_{i\in I}\Phi(i) \to \operatorname{colim}\Phi$ is surjective. Since the evaluation functors preserve colimit, we have the following result.

Lemma 6.2.3 [`SSet.Ω.PreservesLimit.jointly_surjective`]**.** *Let* $\Phi : I \Rightarrow \mathrm{sSet}$ *be a diagram of simplicial sets with colimit. For all* $n \in \mathbb{N}$*,* $x \in (\operatorname{colim}\Phi)_n$*, there exists* $i \in I$ *and* $y \in \Phi(i)_n$ *such that* $v_i(y) = x$ *where* v_i*'s are the legs of the colimit.*

Applying Lemma 6.2.3 to F (note that F is diagram of the opposite category of sSet, that's why there is a switch of limit / colimit), we choose i and y for every $x \in (\lim F)_n$. Then the isomorphism $v_i^* f \cong v_i^* g$ gives us an order isomorphism $f^{-1}(x) \cong g^{-1}(x)$. It follows from the uniqueness of the order isomorphism that the order isomorphisms are independent of the choice of i and y. Also, if $f^{-1}(\varphi(x)) \cong g^{-1}(\varphi(x))$ is obtained from $v_i^* f^{-1}(\varphi(x)) \cong v_i^* g^{-1}(\varphi(x))$, then it's obviously compatible. Hence, by independence, the construction is compatible and we finish the proof of injectivity. The formalization is straightforward. The only thing to notice is that we need `Classical.choice` to define the isomorphism like most of the formalizations in Lean.

Surjectivity: Let $f \in \lim(\mathbf{W}_\alpha \circ F)$. Let $f_i : Y_i \to Fi$ be a representative of $u_i(f) \in \mathbf{W}_\alpha(Fi)$. We want to construct a α-small well-ordered morphisms $g : Y \to \lim F$ whose image is f under Ψ.

For each $x \in (\lim F)_n$, choose i and y using Lemma 6.2.3 such that $v_i(y) = x$ and let $Y_x := f_i^{-1}(y) \subset Y_{i,n}$. We argue that the resulting Y_x is independent of choice of i and x.

For a diagram of sets $\Phi : I \Rightarrow \mathrm{Set}$, we define a relation on the pair $\{(i, x) | i \in I, x \in \Phi i\}$ by $(i, x) \rightsquigarrow (j, y)$ if there exists a morphism $\varphi : i \to j$ such that $\Phi(\varphi)(x) = y$. We denote the equivalence closure of this relation by $\approx$. This is one step in proving every small diagram of sets has colimit. In particular, we have the following lemma

Lemma 6.2.4. *Let* $\Phi : I \Rightarrow \mathrm{Set}$ *be a diagram of sets with colimit. Let* $c_i : \Phi(i) \to \operatorname{colim}\Phi$ *denote the legs of the cocone. For* $i, j \in I$*,* $x \in \Phi(i)$ *and* $y \in \Phi(j)$*, we have* $c_i(i, x) = c_j(j, y)$ *if and only if* $(i, x) \approx (j, y)$*.*

Again, since the evaluation functors preserve colimit, we have a corresponding result on simplicial sets.

Lemma 6.2.5 [`SSetΩ.PreservesLimit.eqvGen_of_app_eq`]**.** *Let* $\Phi : I \Rightarrow \mathrm{sSet}$ *be a diagram of simplicial sets with colimit. Let* $c_i : \Phi(i) \to \operatorname{colim}\Phi$ *denote the legs of the cocone. For* $i, j \in I$*,* $x \in \Phi(i)_n$ *and* $y \in \Phi(j)_n$*, we have* $c_i(i, x) = c_j(j, y)$ *if and only if* $(i, x) \approx (j, y)$*.*

Induction on $\approx$ shows that if $(i, x) \approx (j, y)$, then $f_i^{-1}(x) \cong f_j^{-1}(y)$. Hence, the independence of choices is proved.

Define Y_n to be the disjoint union of Y_x over $x \in (\lim F)_n$. Now we want to equip Y with a simplicial structure. The argument is similar to the one for injectivity. Given

$y \in f_i^{-1}(x) \subset Y_n$ and $\varphi : n \to m$, we want to define $\varphi y \in Y_m$. This is clear if we identify $y \in Y_{i,n}$, we just take $\varphi y \subset Y_{i,m}$. This is valid due to the independence of choice.

However, in formalization, we have to make clear of the informal use of "identification". This is what makes the related Lean code especially lengthy. For example, when defining the simplicial structure of Y_n,

```
map {n m} φ := by
  intro z
  let H := FibreOrderIsoOfAppEq c hc f
    (choose_x hc (φ ~ z.fst)) (φ ~ choose_x hc z.fst)
    (by rw [hom_naturality_apply, choose_spec_x, choose_spec_x]; rfl)
  exact ⟨c.pt.unop.map φ z.fst, H.symm (move φ z.snd)⟩
```

we need to explicitly point out the "identification" `FibreOrderIsoOfAppEq`.

The unavoidable use of `FibreOrderIsoOfAppEq` makes the seemingly easy proof of functor axioms `map_id` and `map_comp` rather complicated, where we have to repeatedly invoke the uniqueness of order isomorphisms. We omit the details here.

It remains to prove $\Psi(g) = f$. We recall another lemma about the category of sets.

Lemma 6.2.6. *Let $\Phi : I \Rightarrow \mathrm{Set}$ be a diagram of sets with limit. Let $c_i : \lim \Phi \to \Phi(i)$ denote the legs of the cone. Let $x, y \in \lim \Phi$. Then $x = y$ if and only if $c_i x = c_i y$ for all $i \in I$.*

Therefore it suffices to show $u_i(\Psi(g)) = u_i(f)$ for all i, which is again a problem of constructing isomorphisms of well-ordered morphisms. Note that the equality is precisely $\mathbf{W}_\alpha(v_i)(g) = f_i$. For $x \in (Fi)_n$, $\mathbf{W}_\alpha(v_i)(g)^{-1}(x) \cong g^{-1}(v_i x) \cong f_i^{-1}(x)$. The first isomorphism follows from the property of pullback (the first remark under Definition 6.2.1) and the second one is by definition of g. With these ingredients, we are ready to define `Pieces`. However, the challenge lies in proving their compatibility, which resembles the situation with functor axioms mentioned earlier. We again omit the details.

6.2.2 The simplicial sets W_α and $\tilde{W}_\alpha$

Definition 6.2.7 [`SSet.W`]**.** *Define the simplicial set W_α as the composition*

$$\Delta^{op} \overset{y^{op}}{\Rightarrow} \mathrm{sSet}^{op} \overset{\mathbf{W}_\alpha}{\Rightarrow} \mathrm{Set}$$

Theorem 6.2.8 [`SSet.Ω.Corepresentable`]**.** *$\mathbf{W}_\alpha$ is corepresented by W_α, i.e., there is an natural isomorphism of functors*

$$\mathbf{W}_\alpha \cong \mathrm{Hom}_{\mathrm{sSet}}(-, W_\alpha)$$

Proof. By def,

$$\mathrm{Hom}(\Delta^n, W_\alpha) \cong (W_\alpha)_n \overset{def}{=} \mathbf{W}_\alpha(\Delta^n).$$

Recall, every simplicial set X is the colimit of its category of elements denoted by X^{el} (see the remark below). Using Theorem 6.2.2 and the fact that $\mathrm{Hom}(-, W_\alpha)$

preserves limits, we have

$$\begin{aligned}\mathrm{Hom}(X, W_\alpha) &\cong \lim \mathrm{Hom}(\Delta^n, W_\alpha)\\ &= \lim \mathbf{W}_\alpha(\Delta^n)\\ &\cong \mathbf{W}_\alpha(\operatorname{colim} \Delta^n)\\ &\cong \mathbf{W}_\alpha(X)\end{aligned}$$

□

Remark. For any presheaf $P : \mathrm{Set}^J$, its category of elements P^{el} is defined as follows:

- The objects are the pair (i, x) where $x \in Pi$.
- The morphism between (i, x) and (j, y) are the morphisms $f : i \to j$ such that $(Pf)(x) = y$.

A diagram of presheaves $F : P^{el,op} \Rightarrow \mathrm{Set}^J$ is given as the composition

$$P^{el,op} \overset{\pi}{\Rightarrow} J^{op} \overset{y}{\Rightarrow} \mathrm{Set}^J$$

where π is the obvious forgetful functor. Then P is the colimit of F.

This result has already been formalized in Mathlib. But similar to the formalization of Theorem 4.1.3, it does not directly apply to `sSet.{u}`, whose codomain lies in a higher universe `Type u`. Therefore, we need to modify the definition by postcomposing `yoneda` with `uliftFunctor`.

```
def functorToRepresentables (X : SSet.{u}) :
    X.Elementsᵒᵖ ⥤ SSet.{u} :=
  (CategoryOfElements.π X).leftOp ⋙ (yoneda ⋙ uliftFunctor)
```

The related results need also be slighted adjusted.

Instead of a natural isomorphism of functors, corepresentability in Mathlib is defined concretely as an equivalence of types with naturality:

```
structure CorepresentableBy (F : C ⥤ Type v) (X : C) where
 homEquiv {Y : C} : (X ⟶ Y) ≃ F.obj Y
 homEquiv_comp {Y Y' : C} (g : Y ⟶ Y') (f : X ⟶ Y) :
  homEquiv (f ≫ g) = F.map g (homEquiv f)
```

Therefore, the proof of Theorem 6.2.8 in Lean is precisely a composition of the three natural equivalences presented in the proof. We elaborate on the details for the first equivalence.

For a limit-preserving functor $G : \mathrm{sSet}^{op} \Rightarrow D$, we have the following result due to uniqueness of limit up to isomorphisms.

```
def SSet.IsoOfPreservesLimit {D : Type w} [Category D]
  (G : SSetᵒᵖ ⥤ D) (Y : SSet)
  [HasLimit (Y.functorToRepresentables.op ⋙ G)]
  [PreservesLimit Y.functorToRepresentables.op G] :
    G.obj (op Y) ≅ limit (Y.functorToRepresentables.op ⋙ G)
```

Note that $f : Y \to X$ induces a natural transformation $X^{el,op} \ggg G \to Y^{el,op} \ggg G$. Hence the limit at the right, as a functor on Y, maps $f : Y \to X$ to the following

```
def IsoOfPreservesLimit_comp (G : SSetᵒᵖ ⥤ D) (X Y : SSet) (f : Y ⟶ X) :
  limit (X.functorToRepresentables.op ⋙ G) ⟶
    limit (Y.functorToRepresentables.op ⋙ G) :=
  limit.pre _ (CategoryOfElements.map f).op.op
```

The naturality is described in the following lemma,

```
lemma IsoOfPreservesLimit_comp_hom :
  G.map (op f) ≫ (IsoOfPreservesLimit G Y).hom =
    (IsoOfPreservesLimit G X).hom ≫ (IsoOfPreservesLimit_comp G f)
```

Applying these to $\mathrm{Hom}(-, W_\alpha)$, we obtain

```
def Ω.CorepresentableAux₁.{u} (α : Cardinal.{u}) (Y : SSet) :
  (Y ⟶ W α) ≃ limit (Y.functorToRepresentables.op ⋙ yoneda.obj (W α))
```

and its naturality.

Definition 6.2.9 [`SSet.Ω.toHom`]. *Every equivalence class f of α-small well-ordered morphisms $Y \to X$ corresponds to a map $X \to W_\alpha$ via the isomorphism in Theorem 6.2.8, We denote it by $\ulcorner f \urcorner$.*

Given an α-small well-ordered morphism $f : Y \to X$, we also denote $\ulcorner f \urcorner$ for the map corresponded to the equivalence class of f by abusing the notation.

Definition 6.2.10 [`SSet.Ω.toObj`]. *Every map $f : X \to W_\alpha$ corresponds to an equivalence class of α-small well-ordered morphisms $Y \to X$ via the isomorphism in Theorem 6.2.8, We denote it by $\llcorner f \lrcorner$.*

Definition 6.2.11 [`SSet.UniSmallWO`]. *Applying the isomorphism in Theorem 6.2.8 to* $\mathrm{id} : W_\alpha \to W_\alpha$ *yields a α-small well-ordered morphism $q_\alpha : \widetilde{W}_\alpha \to W_\alpha$.*

Remark. We can have an explicit characterization of $\widetilde{W}_\alpha$.

1. Under

$$\mathrm{Hom}(W_\alpha, W_\alpha) \cong \mathrm{Hom}(\mathrm{colim}(\Delta^n), W_\alpha) \cong \lim \mathrm{Hom}(\Delta^n, W_\alpha) \cong \lim \mathbf{W}_\alpha(\Delta^n),$$

   id is sent to isomorphism classes $(f_x : X \to \Delta^n | x : \Delta^n \to W_\alpha)$.

2. According to the proof of surjectivity for Theorem 6.2.2, under

$$\lim \mathbf{W}_\alpha(\Delta^n) \cong \mathbf{W}_\alpha(W_\alpha),$$

   $(f_x : X \to \Delta^n | x : \Delta^n \to W_\alpha)$ is sent to a well-ordered morphism $Y \to W_\alpha$, where Y is consisted of copies of $(f : Y \to \Delta^n, s \in f^{-1}(1_{[n]}))$ ranging over the simplexes of W_α.

In conclusion, the fibre of $x \in (W_\alpha)_n = \mathbf{W}_\alpha(\Delta^n)$ along $\widetilde{W}_\alpha \to W_\alpha$ is a copy of the fibre of $1_{[n]}$ along a representative of x (in the form of $Y \to \Delta^n$ for some Y).

$q_\alpha : \widetilde{W}_\alpha \to W_\alpha$ is formalized as follows. Note that, we need to use `Quotient.out` to choose a representative of the equivalence class, which is usually ignored in informal writing as in Definition 6.2.11.

```
abbrev UniSmallWO₀ := Ω.toObj (𝟙 (W α)) -- the equivalence class

abbrev UniSmallWO := Quotient.out $ (equivShrink (Ω_obj₀ α (W α))).symm
    (UniSmallWO₀ α) -- the representative

abbrev W' := (UniSmallWO α).of -- the domain
```

Proposition 6.2.12 [SSet.UniSmallWO.universal]**.** *$q_\alpha : \widetilde{W}_\alpha \to W_\alpha$ is strictly universal for α-small well-ordered morphisms, i.e., any α-small well-ordered can be realized uniquely as a pullback of p.*

Proof. Let $f : X \to W_\alpha$ be a morphism. By the naturality, the diagram commutes:

$$\begin{tikzcd}
\mathrm{Hom}(W_\alpha, W_\alpha) \arrow[r, "\cong"] \arrow[d, "-\circ f"'] & \mathbf{W}_\alpha(W_\alpha) \arrow[d, "f^*"] \\
\mathrm{Hom}(X, W_\alpha) \arrow[r, "\cong"'] & \mathbf{W}_\alpha(X)
\end{tikzcd}$$

So,

$$\begin{tikzcd}
\mathrm{id} \arrow[r, mapsto] \arrow[d, mapsto] & p : \widetilde{W}_\alpha \to W_\alpha \arrow[d, mapsto] \\
f \arrow[r, mapsto] & f^*p
\end{tikzcd}$$

Hence $\ulcorner f^*p \urcorner = f$. By bijectivity, given $g : Y \to X$ a well-ordered morphism, $g = \ulcorner g \urcorner^* p$. □

Proposition 6.2.13 [SSet.UniSmallWO.weaklyUniversal]**.** *$\widetilde{W}_\alpha \to W_\alpha$ is weakly universal for α-small morphisms. (weakly = not neccessarily unique)*

Proof. Every map of simplicial sets can be well-ordered, since every set can be well-ordered. □

6.2.3 The functor $\mathbf{U}_\alpha$, the simplicial sets U_α and $\tilde{U}_\alpha$

Definition 6.2.14 [SSet.Υ]**.** *Let $\mathbf{U}_\alpha \subset \mathbf{W}_\alpha$ be the subobject consisting of isomorphism classes of α-small well-order Kan fibrations.*

Here, the subobject simply means $\mathbf{U}_\alpha(X) \subset \mathbf{W}_\alpha(X)$ for all X. Hence there is a natural transformation $\mathbf{U}_\alpha \hookrightarrow \mathbf{W}_\alpha$ given by inclusions.

Definition 6.2.15 [SSet.U]**.** *Similarly, define U_α to be the composition*

$$\Delta^{op} \overset{y^{op}}{\Rightarrow} \mathrm{sSet} \overset{\mathbf{U}_\alpha}{\Rightarrow} \mathrm{Set}\,.$$

Definition 6.2.16 [SSet.U.toW]**.** *There is a morphism $i : U_\alpha \to W_\alpha$ given by (set map) $i_n : \mathbf{U}_\alpha(\Delta^n) \hookrightarrow \mathbf{W}_\alpha(\Delta^n)$.*

Now we are ready to define the key construction of the universe.

Definition 6.2.17 [`SSet.UniSmallWOKan`]. *Define $p_\alpha : \tilde{U}_\alpha \to U_\alpha$ to be the pullback of p along $U_\alpha \to W_\alpha$.*

Lemma 6.2.18 [`SSet.U.Kan_pullback_snd_simplex`]. *For any $f : \Delta^n \to U_\alpha$, f^*p_α is a Kan fibration.*

Proof. Since p_α is defined as a pullback, the outer square is also a pullback

$$\begin{tikzcd}
\bullet \arrow[r] \arrow[d, "f^*p_\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U}_\alpha \arrow[r] \arrow[d, "p_\alpha"] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{W}_\alpha \arrow[d, "q_\alpha"] \\
\Delta^n \arrow[r, "f"] & U_\alpha \arrow[r] & W_\alpha
\end{tikzcd}$$

Hence $f^*p_\alpha = (i \circ f)^*q_\alpha = \llcorner i \circ f \lrcorner = i_n \llcorner f \lrcorner \in \mathbf{U}_\alpha(\Delta^n)$. It follows that f^*p_α is a Kan fibration by definition of $\mathbf{U}_\alpha$. □

Lemma 6.2.19 [`SSet.UniSmallWOKan.Kan`]. *p_α is a Kan fibration.*

Proof. Let there be a commutative diagram

$$\begin{tikzcd}
\Lambda^n_k \arrow[r] \arrow[d, hook] & \tilde{U}_\alpha \arrow[d, "p_\alpha"] \\
\Delta^n \arrow[r, "f"'] & U_\alpha
\end{tikzcd}$$

By the universal property of pullbacks,

$$\begin{tikzcd}
\Lambda^n_k \arrow[r] \arrow[d, hook] & f^*\tilde{U}_\alpha \arrow[r] \arrow[d, "f^*p_\alpha"] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{U}_\alpha \arrow[d, "p_\alpha"] \\
\Delta^n \arrow[r, "="'] & \Delta^n \arrow[r, "f"'] & U_\alpha
\end{tikzcd}$$

By Lemma 6.2.18, f^*p_α is a Kan fibration, so the left square admits a lift and so does the outer square. □

Lemma 6.2.20 [`SSet.Ω_obj.Kan_iff_factor`]. *An α-small well-ordered morphism $f : Y \to X$ is a fibration iff $\ulcorner f \urcorner : X \to W_\alpha$ factors through U_α.*

Proof. $\Rightarrow$ Assume $f : Y \to X$ is a fibration. Then the pullback of f along any simplex $x : \Delta^n \to X$ is a fibration and $\ulcorner f \urcorner(x) = \ulcorner x^*f \urcorner \in U_\alpha$. Hence $\ulcorner f \urcorner$ factors through U_α.

$\Leftarrow$ The factorisation gives

$$\begin{tikzcd}
Y \arrow[dr, dashed] \arrow[drr] \arrow[dd, "f"] \arrow[ddr, phantom, "\lrcorner", very near start] & & \\
& \tilde{U}_\alpha \arrow[r] \arrow[d, "p_\alpha"] \arrow[dr, phantom, "\lrcorner", very near start] & \tilde{W}_\alpha \arrow[d] \\
X \arrow[r] & U_\alpha \arrow[r] & W_\alpha
\end{tikzcd}$$

It follows that the left square is a pullback. Since p_α is a fibration, so is f. □

Lemma 6.2.21 [`SSet.Υ.Corepresentable`]. *$\mathbf{U}_\alpha$ is corepresented by U_α.*

Proof. For any X, $\mathbf{U}_\alpha(X) \hookrightarrow \mathbf{W}_\alpha(X) \cong \mathrm{Hom}(X, W_\alpha)$ has image $\mathrm{Hom}(X, U_\alpha)$, by Lemma 6.2.20. Hence we have a bijection $\mathbf{U}_\alpha(X) \cong \mathrm{Hom}(X, U_\alpha)$. The naturality follows from the naturality of W_α □

Theorem 6.2.22 [`SSet.UniSmallWOKan.universal`]. *p_α is strictly universal for α-small well-ordered Kan fibrations.*

Proof. Follows from the corepresentability of $\mathbf{U}_\alpha$ as the proof of Proposition 6.2.12. □

Theorem 6.2.23 [`SSet.UniSmallWOKan.weaklyUniversal`]. *p_α is weakly universal for α-small Kan fibrations*

Proof. As in Proposition 6.2.13, every map of simplicial sets can be well-ordered. □

6.3 Logical structures

By Theorem 4.1.3, sSet is locally cartesian closed. We choose a pullback for every map $X \to U_\alpha$ along $p_\alpha : \tilde{U}_\alpha \to U_\alpha$ and define a structure of universe (Definition 5.1.1) on sSet [`SSet.Uni`]. sSet_{U_α} is then our simplicial model [`SSet.Model`]. The choice of pullbacks does not matter, as the coherence issues is auotomatically solved by the construction of sSet_{U_α}.

For sSet_{U_α} to be a model of HoTT, we need to prove

1. it has all the logical structures
2. it validates the univalence axiom

We have fully formalized the results about Π-types. By Theorem 5.2.4, it suffices to provide a Π-structure (Definition 5.2.3) on sSet_{U_α}.

Recall $\alpha_{\mathsf{gen}} : A_{\mathsf{gen}} \to U_\alpha^\Pi$ and $\beta_{\mathsf{gen}} : B_{\mathsf{gen}} \to A_{\mathsf{gen}}$ are both defined to be pullbacks of p_α and hence are Kan fibrations (See Definition 5.2.1). They are also α-small, because α-smallness is stable under pullback [`SSet.SmallFibre.stableUnderPullback`] and p_α is α-small by design.

Therefore, if we have

1. Kan fibrations are stable under the dependent product of along a Kan fibration. In other words, let f, g be a Kan fibration, then $\Pi_f g$ is also a Kan fibration.
2. α-smallness is stable under the dependent product along a α-small map.

then $\Pi_{\alpha_{\mathsf{gen}}}\beta_{\mathsf{gen}} : \Pi_{\alpha_{\mathsf{gen}}} B_{\mathsf{gen}} \to U_\alpha^\Pi$ is an α-small Kan fibration. By Theorem 6.2.23, there is a map $\Pi : U_\alpha^\Pi \to U_\alpha$ such that $\Pi_{\alpha_{\mathsf{gen}}}\beta_{\mathsf{gen}}$ is the pullback, or $\Pi_{\alpha_{\mathsf{gen}}}\beta_{\mathsf{gen}} \cong \Pi^* p_\alpha$, precisely the desired structure.

The first point is easy if we assume the knowledge from Quillen-Kan model structure of simplcial sets. It is known that Kan fibrations and trivial cofibrations form a weak factorization system. Also, Quillen-Kan model structure is right proper, i.e., trivial cofibrations are stable under pullback along Kan fibrations. Then the first point follows from Quillen adjuction. Unfortunately, neither of these classical results has been fully formalized yet, due to the extensive prerequisites required, particularly in simplicial homotopy theory.

Fot the second point, we need some assumptions on α. Recall a cardinality α is

1. regular if it is infinite and it equals its own cofinality.
2. a strong limit if it is not zero and it is closed under powersets
3. inaccessable if uncountable, regular and a strong limit.

We list some of useful properties. We use # to denote the cardinality of a set.

1. (`Cardinal.pow_lt_of_isStrongLimit`) If α is a strong limit caridnal and $\beta, \gamma < \alpha$, then $\beta^\gamma < \alpha$.
2. (`Cardinal.prod_lt_bound_pow_of_lt_of_lt`) Let α be a regular and strong limit caridnal. For a set of sets $\{S_i | i \in I\}$, if $\#I < \alpha$ and for all i, $\#S_i < \alpha$, then $\# \sqcup_i S_i < \alpha^\# I$.

Lemma 6.3.1 [`SSet.SmallFibre.stableUnderPushforward`]. *Let α be inaccessable. Then α-smallness is stable under the dependent product along a α-small map.*

Proof. Let $f : X \to Y$ and $g : Y \to Z$ be α-small. Let $z \in Z_n$. There is an obvious bijection

$$\Pi_g f^{-1}(z) \cong \mathrm{Hom}_{\mathrm{sSet}/Z}(z, \Pi_g f)$$

where we identify z with the map $\Delta^n \to Z$ via Yoneda. By adjuction $g^* \dashv \Pi_g$, we have

$$\mathrm{Hom}_{\mathrm{sSet}/Z}(z, \Pi_g f) \cong \mathrm{Hom}_{\mathrm{sSet}/Y}(g^* z, f).$$

$\mathrm{Hom}_{\mathrm{sSet}/Y}(g^* z, f)$ injects into

$$\bigsqcup_{k \in \mathbb{N}} \bigsqcup_{y \in g^{-1}(z(\Delta^n_k))} \{h | h : (g^* z)^{-1}(y) \to f^{-1}(y)\}.$$

Fix k and $y \in g^{-1}(z(\Delta^n_k))$. $\#(g^* z)^{-1}(y) < \alpha$ since $(g^* z)^{-1}(y) \cong z^{-1}(g(y))$ and Δ^n_k is finite. By smallness of f, $\#f^{-1}(y) < \alpha$. Therefore $\#\{h | h : (g^* z)^{-1}(y) \to f^{-1}(y)\} < \alpha$, by strong limit.

Fix k, $\#g^{-1}(z(\Delta^n_k)) < \alpha$, since g is α-small, $z(\Delta^n_k)$ is finite and α is infinite.

Finally it follows from regularity, strong limit and $\#\mathbb{N} < \alpha$ that,

$$\# \bigsqcup_{k \in \mathbb{N}} \bigsqcup_{y \in g^{-1}(z)} \{h | h : (g^* z)^{-1}(y) \to f^{-1}(y)\} < \alpha$$

□

Chapter 7

Conclusion

This thesis has focused on formalizing the core theory of the simplicial model in the context of Homotopy Type Theory (HoTT). A significant point achieved in this work is the formalization of the simplicial model's Π-structure (modulo a theorem from the Kan-Quillen structures of simplicial sets), which successfully demonstrates its ability to model dependent types.

While the formalization of most of the other structures, such as Σ-types, will follow a similar approach, Id-types and universes pose challenges. Unfortunately, were unable to address them within the limited time frame of this thesis. Additionally, we would hope to formalize the proof that the simplicial model validates the univalence axiom. However, this goal also remain out of reach due to the limited time and the extensive prerequisites required from the theories of simplicial sets and model categories.

Despite these limitations, this thesis has established a solid foundation for further formalization of the simplicial model. In future work, we look forward to formalizing the remaining structures, incorporating the prerequisites needed for univalence, and finally proving it. This effort not only deepens our understanding of the simplicial model and univalence but also reinforces the role of formal proof assistants like Lean in advancing mathematical research.

Appendix A

Type theory

A.1 Martin-Löf type theory

By a Martin-Löf type theory, we mean a type theory with

1. judgments : $\Gamma\ \mathsf{ctx}$, $\Gamma \vdash A\ \ \mathsf{type}$, $\Gamma \vdash A = B\ \ \mathsf{type}$, $\Gamma \vdash a : A$, $\Gamma \vdash a = b : A$.
2. the structural rules in A.1.1
3. a selection of logical rules from A.1.2

A.1.1 Structural rules

$$\frac{}{\mathsf{ctx}}\ \textsf{ctx-EMP} \qquad \frac{x_1{:}A_1, ..., x_{n-1}{:}A_{n-1} \vdash x_n : A_n}{(x_1{:}A_1, ..., x_n{:}A_n)\ \mathsf{ctx}}\ \textsf{ctx-EXT}$$

$$\frac{(x_1{:}A_1, ..., x_n{:}A_n)\ \mathsf{ctx}}{x_1{:}A_1, ..., x_n{:}A_n \vdash x_i : A_i}\ \text{VBLE}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash \mathcal{J}}{\Gamma, \Delta[a/x] \vdash \mathcal{J}[a/x]}\ \text{SUBST} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, \Delta \vdash \mathcal{J}}{\Gamma, x{:}A, \Delta \vdash \mathcal{J}}\ \text{WKG}$$

$$\frac{\Gamma \vdash A\ \ \mathsf{type}}{\Gamma \vdash A = A\ \ \mathsf{type}}\ \textsf{type-REFL} \qquad \frac{\Gamma \vdash A = B\ \ \mathsf{type}}{\Gamma \vdash B = A\ \ \mathsf{type}}\ \textsf{type-SYMM}$$

$$\frac{\Gamma \vdash A = B\ \ \mathsf{type} \qquad \Gamma \vdash B = C\ \ \mathsf{type}}{\Gamma \vdash A = C\ \ \mathsf{type}}\ \textsf{type-TRAN}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a = a : A}\ \textsf{term-REFL} \qquad \frac{\Gamma \vdash a = b : A}{\Gamma \vdash b = a : A}\ \textsf{term-SYMM}$$

$$\frac{\Gamma \vdash a = b : A \qquad \Gamma \vdash b = c : A}{\Gamma \vdash a = c : A}\ \textsf{term-TRAN}$$

A.1.2 Logical rules

Π-types

$$\frac{\Gamma, x{:}A \vdash B \;\; \mathsf{type}}{\Gamma \vdash \Pi_{x:A} B \;\; \mathsf{type}}\;\Pi\text{-}\textsc{form} \qquad \frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda_{x:A} b : \Pi_{x:A} B \;\; \mathsf{type}}\;\Pi\text{-}\textsc{intro}$$

$$\frac{\Gamma \vdash f : \Pi_{x:A} B \;\; \mathsf{type} \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\;\Pi\text{-}\textsc{elim}$$

$$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash \lambda_{x:A} b(x) \cdot a = b[a/x] : B[a/x]}\;\Pi\text{-}\textsc{comp}$$

Σ-types

$$\frac{\Gamma, x{:}A \vdash B \;\; \mathsf{type}}{\Gamma \vdash \Sigma_{x:A} B \;\; \mathsf{type}}\;\Sigma\text{-}\textsc{form} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : \Sigma_{x:A} B \;\; \mathsf{type}}\;\Sigma\text{-}\textsc{intro}$$

$$\frac{\Gamma, z{:}\Sigma_{x:A} B \vdash C \;\; \mathsf{type} \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \qquad \Gamma \vdash p : \Sigma_{x:A} B}{\Gamma \vdash \mathsf{ind}_{\Sigma_{x:A} B}(C, g, p) : C[p/z]}\;\Sigma\text{-}\textsc{elim}$$

$$\frac{\begin{array}{c}\Gamma, z{:}\Sigma_{x:A} B \vdash C \;\; \mathsf{type} \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}_{\Sigma_{x:A} B}(C, g, (a,b)) = g[a,b/x,y] : C[(a,b)/z]}\;\Sigma\text{-}\textsc{comp}$$

Id-types

$$\frac{\Gamma \vdash a : A \;\; \mathsf{type} \qquad \Gamma \vdash b : A \;\; \mathsf{type}}{\Gamma \vdash \mathsf{Id}_A(a,b) \;\; \mathsf{type}}\;\mathsf{Id}\text{-}\textsc{form} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : \mathsf{Id}_A(a,a) \;\; \mathsf{type}}\;\mathsf{Id}\text{-}\textsc{intro}$$

$$\frac{\begin{array}{c}\Gamma, x{:}A, y{:}A, p : \mathsf{Id}_A(x,y) \vdash C \;\; \mathsf{type} \qquad \Gamma, z{:}A \vdash c : C[z,z,\mathsf{refl}_z/x,y,p] \\ \Gamma \vdash q : \mathsf{Id}_A(a,b)\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathsf{Id}_A}(C, c, q) : C[a,b,q/x,y,p]}\;\mathsf{Id}\text{-}\textsc{elim}$$

$$\frac{\begin{array}{c}\Gamma \vdash a : A \\ \Gamma, x{:}A, y{:}A, p : \mathsf{Id}_A(x,y) \vdash C \;\; \mathsf{type} \qquad \Gamma, z{:}A \vdash c : C[z,z,\mathsf{refl}_z/x,y,p]\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathsf{Id}_A}(C, c, \mathsf{refl}_a) = c[a/z] : C[a,a,\mathsf{refl}_a/x,y,p]}\;\mathsf{Id}\text{-}\textsc{comp}$$

Empty type 0

$$\frac{\Gamma \;\; \mathsf{ctx}}{\Gamma \vdash \mathsf{0} \;\; \mathsf{type}}\;\mathsf{0}\text{-}\textsc{form} \qquad \frac{\Gamma, x{:}\,\mathsf{0} \vdash C \;\; \mathsf{type}}{\Gamma, x{:}\,\mathsf{0} \vdash \mathsf{ind}_{\mathsf{0}}(C) : C}\;\mathsf{0}\text{-}\textsc{elim}$$

Unit type 1

$$\frac{\Gamma \;\; \mathsf{ctx}}{\Gamma \vdash \mathsf{1} \;\; \mathsf{type}}\;\mathsf{1}\text{-}\textsc{form} \qquad \frac{\Gamma \;\; \mathsf{ctx}}{\Gamma \vdash \star : \mathsf{1}}\;\mathsf{1}\text{-}\textsc{form}$$

$$\frac{\Gamma, x{:}\,\mathsf{1} \vdash C \;\; \mathsf{type} \qquad \Gamma \vdash d : C[\star/x]}{\Gamma, x{:}\,\mathsf{1} \vdash \mathsf{ind}_{\mathsf{1}}(d) : C}\;\mathsf{1}\text{-}\textsc{elim}$$

$$\frac{\Gamma, x{:}\,\mathsf{1} \vdash C \ \ \mathsf{type} \qquad \Gamma \vdash d : C[\star/x]}{\Gamma \vdash \mathsf{ind}_{\mathsf{1}}(d)[\star/x] = d : C[\star/x]} \ \mathsf{1}\text{-}\textsc{comp}$$

Inner universe U and El

$$\frac{}{\vdash \mathsf{U} \ \ \mathsf{type}} \qquad \frac{}{x{:}\,\mathsf{U} \vdash \mathsf{El}(x) \ \ \mathsf{type}}$$

U may have certain logical structures, such as Π-types:

$$\frac{\Gamma \vdash a : \mathsf{U} \qquad \Gamma, x{:}\,\mathsf{El}(a) \vdash b : \mathsf{U}}{\Gamma \vdash \pi(a,b) : \mathsf{U}}$$

The inner Π-types π is said to be closed if the following rule holds:

$$\frac{\Gamma \vdash a : \mathsf{U} \qquad \Gamma, x{:}\,\mathsf{El}(a) \vdash b : \mathsf{U}}{\Gamma \vdash \mathsf{El}(\pi(a,b)) = \Pi_{x:\mathsf{El}(a)}\,\mathsf{El}(b) \ \ \mathsf{type}}$$

Similar for other logical rules.

A.1.3 The univalence axiom

If $\Gamma, x{:}A \vdash B \ \ \mathsf{type}$ and B does not depend on A, i.e., there is no occurrence of x in B, we denote $A \to B$ to be $\Pi_{x:A}B$.

Now given $\Gamma \vdash f : A \to B$, we define

1. (Left inverse) $\mathsf{LInv}(f) := \Sigma_{g:B\to A}\Pi_{x:A}\,\mathsf{Id}_A(g(f(x),x))$
2. (Right inverse) $\mathsf{RInv}(f) := \Sigma_{g:B\to A}\Pi_{x:B}\,\mathsf{Id}_B(f(g(x),x))$
3. (H-isomorphism) $\mathsf{isHIso}(f) := \mathsf{LInv}(f) \times \mathsf{RInv}(f)$
4. (Type of h-isomorphisms) $\mathsf{HIso}(A,B) := \Sigma_{f:A\to B}\mathsf{isHIso}(f)$

Suppose $\Gamma \vdash A \ \ \mathsf{type}$. Let $\mathrm{id}_A : A \to A$ be given by $\Gamma, x{:}A \vdash x : A$ via Π-Intro. It is obviously an h-isomorphism with inverses itself. Hence

$$\Gamma, x{:}A \vdash \mathrm{id}_A : \mathsf{HIso}(A,A)$$

Hence via the Id-Elim, we have

$$x, y{:}U, u{:}\,\mathsf{Id}_U(x,y) \vdash w_{x,y,u} : \mathsf{HIso}(\mathsf{El}(x), \mathsf{El}(y)),$$

and again via Π-Intro,

$$x, y{:}U \vdash w_{x,y} : \mathsf{Id}_U(x,y) \to \mathsf{HIso}(\mathsf{El}(x), \mathsf{El}(y)).$$

Finally, the univalence axiom is stated as follows:

Definition A.1.1.
$$x, y{:}U \vdash \mathit{univalence}(x,y) : \mathit{isHIso}(w_{x,y}),$$
where $\mathit{univalence}$ *is a primitive constant.*

A.2 Pure type systems

A.2.1 Definition

Pure type system (PTS) is a form of typed lambda calculus with considerable flexibility on sorts and their relations. (See [11] or [1] Section 5.2)

The specification $\mathbf{S}$ of a PTS consists of three sets

1. $\mathcal{S}$ the set of sorts
2. $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ the set of axioms
3. $\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ the set of relations

Fix $\mathcal{V}$ a countably infinite set of variables. The set $\mathcal{T}$ of pseudoterms is defined using Backus–Naur form as

$$\mathcal{T} = \mathcal{V} \mid \mathcal{S} \mid \mathcal{T}\mathcal{T} \mid \Pi\mathcal{V}{:}\mathcal{T}.\mathcal{T} \mid \lambda V{:}\mathcal{T}.\mathcal{T}.$$

One-step beta reduction $\to_\beta$ is the compatible closure of the relation

$$(\lambda x{:}A.M)N \rightsquigarrow M[N/x]$$

where $[N/x]$ stands for the explicit substitution.

Beta reduction $\twoheadrightarrow_\beta$ is the reflexive symmetric transitive closure of one-step beta reduction $\to_\beta$. Beta conversion $\simeq_\beta$ is the equivalence closure of beta reduction $\twoheadrightarrow_\beta$.

The set $\mathcal{T}$ of pseudocontexts is defined as

$$\mathcal{C} = \cdot \mid \mathcal{C}, \mathcal{V}{:}\mathcal{T}.$$

Define for a pseudocontext Γ, $\operatorname{dom}\Gamma$ the sets of variables occurring in Γ. We say $(x{:}A) \in \Gamma$ if Γ contains $(x{:}A)$.

The set $\mathcal{J}$ of pseudojudgements is defined as

$$\mathcal{J} = \mathcal{C} \vdash \mid \mathcal{C} \vdash \mathcal{T} : \mathcal{T}.$$

The inference rules of PTS is then as follows

$$\frac{}{\cdot \vdash}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash \cdot}\; x \notin \operatorname{dom}\Gamma$$

$$\frac{\Gamma \vdash \cdot}{\Gamma \vdash s_1 : s_2}\; (s_1, s_2) \in A$$

$$\frac{\Gamma \vdash \cdot}{\Gamma \vdash x : A}\; (x : A) \in \Gamma$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3}\; (s_1, s_2, s_3) \in R$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2 \qquad \Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B}\; (s_1, s_2, s_3) \in R$$

$$\frac{\Gamma \vdash f : \Pi x{:}A.B \qquad \Gamma \vdash a : A}{\Gamma \vdash f a : B[a/x]}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A' : s}{\Gamma \vdash a : A'}\ A \simeq_\beta A'$$

The formalization uses de Bruijn index, see `PTS.lean`. In particular, the pseudoterms are formalized as follows. Note that no free variables are mentioned in `pi` and `abs`.

```
inductive PTerm (S : Specification)
| var : ℕ → PTerm S
| sort : S.sort → PTerm S
| app : PTerm S → PTerm S → PTerm S
| pi : PTerm S → PTerm S → PTerm S
| abs : PTerm S → PTerm S → PTerm S
```

We have proven four important meta-properties of PTS, referring to [9].

1. (Confluence) If for pseudoterms A, B we have $A \simeq_\beta B$ then there is a pseudoterm C such that $A \twoheadrightarrow_\beta C$ and $B \twoheadrightarrow_\beta C$.
2. (Weakening) If $\Gamma \vdash A : s$ and $\Gamma, \Delta \vdash b : B$ (plus $x \notin \mathrm{dom}(\Gamma, \Delta)$), then
$$\Gamma, x{:}A, \Delta \vdash b : B.$$
3. (Substitution) If $\Gamma \vdash a : A$ and $\Gamma, x : A, \Delta \vdash b : B$, then
$$\Gamma, \Delta[a/x] \vdash b[a/x] : B[a/x].$$
4. (Subject reduction) If $\Gamma \vdash a : A$ and $a \twoheadrightarrow_\beta a'$ then $\Gamma \vdash a' : A$.

A.2.2 The syntactic category of PTS

The morphisms (explicit substitutions) between two pseudocontexts are defined exactly the same as in Theorem 2.1.1. We define a predicate

```
inductive isMor (Γ : PCtx S) : PCtx S → PCtx S → Type _
| nil : isMor Γ [] []
| cons : isMor Γ Δ F → (Γ ⊢ simulSubst D 0 F : !s) →
      (Γ ⊢ f : (simulSubst D 0 F)) → isMor Γ (D :: Δ) (f :: F)
```

where `simulSubst` is the function of (syntactic) simultaneous substitutions.

Since we are using de Bruijn index, the identity morphism of $x_0{:}A_1, ..., x_n{:}A_n$ is formalized as $(\#0, ..., \#n)$ [`PureTypeSystem.id`]. Also, if Γ is well-formed, then its identity is indeed a morphism

```
def id_isMor (Γ : PCtx S) (h : Γ ⊢ ·) :
   isMor Γ Γ (id Γ)
```

This seemingly trivial result turns out to be difficult to prove, as it requires calculating de Bruijn indices under simultaneous substitutions. Similar challenges arise when proving that the composition of two morphisms is itself a morphism [`PureTypeSystem.pcomp_isMor`].

We define beta conversion for pseudocontexts.

1. One-step beta reduction $\rightarrow_\beta$. For $\Gamma = x_1{:}A_1, ..., x_n{:}A_n$ and $\Gamma' = x_1{:}A'_1, ..., x_m{:}A'_m$, $\Gamma \rightarrow_\beta \Gamma'$ if and only if $m = n$ and there exists i such that $A_i \rightarrow_\beta A'_i$ and $A_j \equiv A'_j$ for $j \neq i$.

2. Beta reduction $\twoheadrightarrow_\beta$ is the reflexive symmetric transitive closure of $\rightarrow_\beta$

3. Beta conversion $\simeq_\beta$ is the equivalence closure of $\twoheadrightarrow_\beta$.

We defined `QCtx` as well-formed contexts modulo $\simeq_\beta$. These are the objects of the desired category.

```
structure Ctx (S : Specification) where
  ctx : PCtx S
  wf : Nonempty (ctx ⊢ ·)

def betac (Γ Δ : Ctx S) : Prop :=
  Nonempty (Γ.ctx ≃β Δ.ctx)

instance setoid (S : Specification) : Setoid (Ctx S) where
  iseqv := betac.equivalence

def QCtx (S : Specification) := Quotient (Ctx.setoid S)
```

Morphisms are also considered modulo $\simeq_\beta$.

```
structure hom₀ (Γ Δ : Ctx S) where
  seq : PCtx S
  is : Nonempty (isMor Γ.ctx Δ.ctx seq)

def betac {Γ Δ : Ctx S} (γ δ : hom₀ Γ Δ) : Prop :=
  Nonempty (γ.seq ≃β δ.seq)

instance setoid (Γ Δ : Ctx S) : Setoid (hom₀ Γ Δ) where
  iseqv := betac.equivalence

def hom (Γ Δ : QCtx S) : Type _ :=
  Quotient (hom₀.setoid Γ.out Δ.out)

instance : Category (QCtx S) where
  Hom := hom
  id := QCtx.id
```

Finally, we prove the data form a contextual category.

```
instance instCategory: Category (QCtx S)

instance instContextualCategory : ContextualCategory (QCtx S)
```

The idea of the proof is described in Theorem 2.1.1. Once again, the main challenge lies in manipulating de Bruijn indices. We omit the details here and refer interested readers to the code for further insights.

Bibliography

[1] Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda calculus with types*. Cambridge University Press, 2013.

[2] Rafaël Bocquet. Strict rezk completions of models of hott and homotopy canonicity. *arXiv preprint arXiv:2311.05849*, 2023.

[3] Guillaume Brunerie. initiality. https://github.com/guillaumebrunerie/initiality/tree/v2.0, 2020.

[4] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of pure and applied logic*, 32:209–243, 1986.

[5] Pierre Clairambault and Peter Dybjer. The biequivalence of locally cartesian closed categories and martin-löf type theories. *Mathematical Structures in Computer Science*, 24(6):e240606, 2014.

[6] Institute for Advanced Study The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Princeton, 2013.

[7] Sina Hazratpour. Poly. https://github.com/sinhp/Poly/blob/c297c860ac70eddab7f8e4c41aae5fbd28edbcde/Poly/LCCC/Basic.lean, 2024.

[8] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after Voevodsky). *Journal of the European Mathematical Society*, 23(6):2071–2126, 2021.

[9] Vincent Siles and Hugo Herbelin. Equality is typable in semi-full pure type systems. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 21–30. IEEE, 2010.

[10] Taichi Uemura. *Abstract and concrete type theories*. PhD thesis, University of Amsterdam, 2021.

[11] Floris van Doorn, Herman Geuvers, and Freek Wiedijk. Explicit convertibility proofs in pure type systems. In *Proceedings of the Eighth ACM SIGPLAN international workshop on Logical frameworks & meta-languages: theory & practice*, pages 25–36, 2013.