INDEXED AND FIBERED PRESENTATIONS OF PRESHEAVES IN TYPE THEORY

And their equivalence via the Grothendieck construction

Moana Laurent JUBERT

Advisor :

Hugo HERBELIN

M2 de Mathématiques fondamentales
Septembre 2022
Abstract

There are two “canonical” ways to define presheaves in type theory: with an indexed presentation, or with a fibered one. Both presentations should be of course equivalent, and the exact nature of this equivalence appears to be related to a well-known result of category theory, the Grothendieck construction. This master’s thesis aims at presenting all the basic tools necessary to understand this equivalence in the special case of direct categories.

Contents

Introduction 1

1 Presheaves 3
  1.1 Definition ........................................... 3
  1.2 Yoneda lemma ........................................ 4
  1.3 Density theorem ...................................... 7
  1.4 Semi-simplicial sets ................................ 8
    1.4.1 Equivalent definitions ............................. 9
    1.4.2 Geometric intuition .............................. 11

2 Grothendieck construction 13
  2.1 Discrete version ...................................... 13
  2.2 Categorical version ................................... 15
  2.3 To infinity and beyond ............................... 19

3 Presheaves in type theory 20
  3.1 Two possible presentations ......................... 20
  3.2 Reedy categories .................................... 22
  3.3 Polygraphs ........................................... 25
  3.4 Transitioning from fibered to indexed ............ 27
  3.5 Conclusion ........................................... 29

4 A “Yoneda-esque” presentation 31
  4.1 Staying at Hom ....................................... 32
  4.2 Definition ........................................... 34
  4.3 Why “Yoneda-esque”? ............................... 35
  4.4 Coherence becomes associativity ................... 39

References 42
Introduction

The initial goal of my internship was to investigate the Grothendieck construction in type theory. Hugo Herbelin have been working for quite some time on the various ways to implement presheaves in type theory, most notably semi-simplicial types [4]. He presented to me the observation that there are at least two “canonical” ways to define presheaves: with an indexed presentation, or with a fibered one. The correspondence between both appears to be tightly related with the Grothendieck construction.

As I come from more “classical” areas of mathematics I was unfamiliar with type theory, and in the end I have spent a considerable amount of my time trying to find a similar distinction on the presentation of presheaves directly at the level of category theory — in the hopes of constructing a clearer picture in my mind. Even though I did not succeed, I have learned a lot on the Grothendieck construction and type theory over the course of my three months at the Institut de recherche en informatique fondamentale (IRIF). In the end, this internship took the turn of making general conjectures about the indexed and fibered presentations of presheaves.

This text is divided in four parts of roughly the same size. I first define presheaves, and make a series of general statements about them. I then move on to introducing the Grothendieck construction in category theory, in its simplest versions. Next, I explain what the indexed and fibered presentations of presheaves are in type theory, in the case of direct categories. I finally present some unfinished work on a variant of the indexed presentation that I call “Yoneda-esque” for its “family resemblance” with the Yoneda lemma, and formulate a number of open questions.

Although it is barely mentioned in this work, homotopy type theory (HoTT) had a special place in my considerations. HoTT is a recent paradigm shift in type theory, with the observation that the equality type $x =_A y$ can be thought as the type of “all paths from $x$ to $y$”. Higher up, $p =_{x =_A y} p'$ can be thought as the type of “all continuous deformations of the path $p$ into the path $p'$”, and so on. In short, types such as $A$ are interpreted as being ∞-groupoids (see [11] for more). As a consequence, what would be interesting is how the higher structure would interact with the different presentations and their correspondence, and how what we already know in higher category theory could help us understand what would be a reasonable construction of presheaves in HoTT.
Acknowledgements

I would like to thank my advisor Hugo Herbelin for his kind support, and for guiding me during my three months as an intern. I would also like to thank Daniel de Rauglaudre with whom I have had the pleasure of being his last “co-bureau”, as he went into retirement at the end of my stay.
1 Presheaves

Presheaves are a key ingredient of category theory. It is important to get them right, as I will make heavy use of presheaves thereafter.

I assume the reader is already familiar with the most basic notions of category theory (e.g. definition, functors, limits and colimits, ...), otherwise a good introduction written by Emily Riehl is available at [13].

1.1 Definition

**Definition.** A presheaf over some category $\mathcal{C}$ is the data of a functor $X$ from the opposite category $\mathcal{C}^{\text{op}}$ to the category of (small) sets $\text{Set}$:

$$X : \mathcal{C}^{\text{op}} \longrightarrow \text{Set}$$

It is standard to write $Xc$ instead of $X(c)$ when applying the presheaf $X$ to some object $c \in \mathcal{C}$. One can think of $X$ as a “generalized set”, where each element is labeled with an object of $\mathcal{C}$. In this case, $Xc$ is the set of elements labeled with $c$.

As $X$ is also a functor, given a morphism $f : c \longrightarrow c'$ of $\mathcal{C}$, the corresponding map $Xf$ sends elements labeled with $c'$ to elements labeled with $c$:

Thus if presheaves are indeed generalized sets, they additionally reflect the structure of the base category $\mathcal{C}$ in a sense that will be soon made precise. A different, albeit completely equivalent point of view is to think of $\mathcal{C}$ as the base space of some projection $\Sigma$ such that the fibers are exactly the labeled elements:

$$\Sigma^{-1}(c) \cong Xc$$
This, again, will be made precise in a moment. The domain \( \text{el} X \) is called the category of elements of \( X \), and every pair \((\text{el} X, \Sigma)\) is uniquely associated to a presheaf \( X \). This bijective correspondence is already a special case of the Grothendieck construction, as we will see in the next section.

The collection of presheaves over a category \( \mathcal{C} \) forms a well-behaved category usually noted \( \text{PSh}(\mathcal{C}) \), or sometimes \( \mathcal{P} \). Morphisms are the natural transformations between presheaves, and it is both complete and cocomplete. A mildly nontrivial fact is that when \( \mathcal{C} \) is locally small, \( \text{PSh}(\mathcal{C}) \) contains a copy of the original category \( \mathcal{C} \) through the Yoneda embedding, which we will now see.

### 1.2 Yoneda lemma

Let us assume that \( \mathcal{C} \) is locally small, that is, for any two objects \( c, c' \in \mathcal{C} \) the collection \( \mathcal{C}(c, c') \) of morphisms from \( c \) to \( c' \) is a (small) set. This gives us a well-defined hom functor into the category of sets:

\[
\mathcal{C}(-, -) : \mathcal{C}^{\text{op}} \times \mathcal{C} \to \text{Set}
\]

It acts contravariantly on the left as pre-composition is contravariant, while post-composition is covariant. Another — perhaps obnoxious — way to formulate this is to say that \( \mathcal{C} \) is a \( \text{Set} \)-enriched category. Now given any object \( c \in \mathcal{C} \), we have:

\[
\mathcal{C}(-, c) : \mathcal{C}^{\text{op}} \to \text{Set}
\]

i.e. to every object of a locally small category, we can associate a canonical presheaf. This is the definition of the Yoneda embedding:

**Definition 2.** The map \( Y : \mathcal{C} \to \text{PSh}(\mathcal{C}) \) defined as:

\[
Y(c) \overset{\text{def}}{=} \mathcal{C}(-, c)
\]

is a well-defined functor, called the Yoneda embedding.

Again, it is standard to write \( Yc \), or even \( \mathcal{C} \) instead of \( Y(c) \). If a presheaf \( X \) is (naturally) isomorphic to some \( \mathcal{C} \) it is said to be representable, and we say that \( X \) is represented by \( c \). The name comes from the fact that it is enough to know how to “probe” the object \( c \) — i.e. to know \( \mathcal{C}(-, c) \) — in order to know \( X \). In some sense, \( c \) then “encodes” everything there is to know about the shape of \( X \).

Of course, most presheaves are not representable. Yet, their structure is completely determined by their relation to the representable ones. This is the
statement of the so-called “Yoneda lemma”:

**Lemma 1 (Yoneda).** $\text{Hom}(\tilde{c}, X) \cong Xc$

**Proof.** We only have to unfold the definition of a natural transformation from $\tilde{c}$ to $X$. The data of a natural transformation $\gamma \in \text{Hom}(\tilde{c}, X)$ is the data of a family of morphisms $\gamma_d : \tilde{c}(d) \toXd$ in Set such that for any arrow $f : d \to d'$ in $\mathcal{C}$, the following diagram commutes:

\[
\begin{array}{ccc}
\tilde{c}(d') & \xrightarrow{\gamma_{d'}} & Xd' \\
\downarrow{\tilde{c}(f)} & & \downarrow{Xf} \\
\tilde{c}(d) & \xrightarrow{\gamma_d} & Xd
\end{array}
\]

The identity $1_c \in \mathcal{C}(c,c)$ is sent to an element of $Xc$ through $\gamma_c$. I now claim that $\gamma$ as a whole is completely determined by $\gamma_c(1_c)$. For any arrow $f : d \to c \in \tilde{c}(d)$, we have:

\[
\begin{array}{ccc}
\mathcal{C}(c,c) & \xrightarrow{\gamma_c} & Xc \\
\downarrow{f^*} & & \downarrow{Xf} \\
\mathcal{C}(d,c) & \xrightarrow{\gamma_d} & Xd
\end{array}
\]

where $f^*$ is defined as the pre-composition by $f$. We then conclude by observing that $\gamma_d(f) = Xf(\gamma_c(1_c))$.

By analogy, one could argue that a natural transformation $\gamma$ from $\tilde{c}$ to $X$ is some sort of “linear map” with $\gamma_c(1_c)$ as its coefficient. We will see it again when discussing the “Yoneda trick” in type theory. For now, what the lemma states is that you only need to look at the “incoming arrows” of a presheaf to know everything about it:

$\tilde{c} \to X$

With the “generalized sets” point of view, it amounts to picking exactly one element labeled with $c$ from $X$. This is not unlike the usual intuition in Set, where looking at the elements of a set $A$, or at the arrows $* \to A$ from any
singleton to $A$, is exactly the same thing. In fact, it is quite easy to show that the category $\text{Set}$ is isomorphic to $\text{PSh}(\ast)$.

I claimed earlier that $\text{PSh}(C)$ contains a copy of the original category $C$. This is an immediate consequence of the Yoneda lemma:

**Lemma 2.** The Yoneda embedding is fully faithful.

Indeed, for any two objects $c, d \in C$ we have that $\text{Hom}(\hat{c}, \hat{d}) \cong C(c, d)$ by the Yoneda lemma. Thus $Y$ truly is an embedding of $C$ into $\text{PSh}(C)$. As presheaves are completely determined by the arrows “coming out of” $Y(C)$ into them, the reader should now easily see how the structure of $C$ and the “internal” structure of a presheaf are tightly related:

\[
\begin{array}{ccc}
\hat{c} & \xrightarrow{f} & \hat{d} \\
\downarrow f & \ & \downarrow \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \Quad...
1.3 Density theorem

**Definition 3.** The *category of elements of* $X$, noted $\text{el} \ X$, is the data of objects $(c, x)$ for every $x \in Xc$, together with morphisms $f : (c, x) \rightarrow (c', y)$ for every $f : c \rightarrow c'$ of $C$ such that $Xf(y) = x$.

$\text{el} \ X$ contains all the elements of $X$ paired with their appropriate label, hence the name “category of elements”. There is a canonical projection $\Sigma : \text{el} \ X \rightarrow C$ such that $\Sigma(c, x) \overset{\text{def}}{=} c$. Then we indeed have that $\Sigma^{-1}(c) \cong Xc$ by definition.

The definition of $\text{el} \ X$ may prompt us to look for a universal property. If $\text{Set}_*$ is the category of pointed sets (i.e. pairs $(a, A)$ with $a \in A$), then $\text{el} \ X$ can be seen as a subcategory of the product $C \times \text{Set}_*$ with objects $(c, (x, Xc))$. This is exactly the data of a pullback square along $X$ (with the appropriate variance) and the evident projection from $\text{Set}_*$ to $\text{Set}$:

```
\[
\begin{array}{ccc}
\text{el} \ X & \overset{\Sigma}{\longrightarrow} & \text{Set}_* \\
\downarrow & & \downarrow \\
C & \overset{X}{\longrightarrow} & \text{Set} \\
\end{array}
\]
```

A remarkable fact of $\Sigma$ is that because $X$ is a presheaf, for any $y \in Xc'$ and any morphism $f : c \rightarrow c'$ of $C$, the arrow $(Xf(y), c) \rightarrow (y, c')$ is the unique lift of $f$ into $\text{el} \ X$. Maps such as $\Sigma$ with this property are called *discrete right fibrations*. As I hinted earlier, this is a first illustration of the Grothendieck construction: *any discrete right fibration is equivalent to the data of a presheaf*.

But for now, we are still only looking at presheaves. The category of elements tells us exactly how the pieces fit together:

**Theorem 1 (Density).** If $C$ is also small, then $X$ is isomorphic to the colimit of the projection $\Sigma$ composed with the Yoneda embedding:

\[
\lim_{\rightarrow} Y\Sigma \cong X
\]

This is usually called the “density theorem”. One way to understand the name is to think of the image of $C$ through the Yoneda embedding as “being dense”, since any presheaf is actually a colimit of representables. The size
condition on $C$ is necessary to make sure that the colimit is small. An idea of proof is the following: glue as many disjoint copies of $c'$s as there are in each $Xc$, then identify what should be equal under the action of the $Xf$'s. This is trivial to state using coends, but it would require us to make an important détour in the realm of abstract nonsense. In any case, an introduction to coends is available in [12] and a complete proof of the previous theorem using Kan extensions instead is available in [13, p. 212].

The somehow “converse” statement is also true: $\mathbf{PSh}(C)$ is the “most general” category containing a copy of $C$ and having all (small) colimits, i.e.:

\begin{quote}
THEOREM 2. $\mathbf{PSh}(C)$ is the free (small) cocompletion of $C$, that is for any functor:

$$F : C \rightarrow D$$

into a cocomplete category $D$, there exists an extension:

$$\exists ! \hat{F}$$

which is unique up to isomorphism.
\end{quote}

The proof is more technical, and again can be found in [13, p. 213] as a remark. What is more interesting is that this theorem provides a completely different characterization of presheaves by a universal property, which might be worth exploring in type theory.

### 1.4 Semi-simplicial sets

I have presented some of the general properties of presheaves so far. I will now focus on the specifics of semi-simplicial sets (also sometimes called $\Delta$-sets in the literature).

Simplicial sets (without the semi-) encode the ways to glue $n$-simplices together along their faces in order to build arbitrarily complex shapes. They are “rich” enough to entirely capture the “algebraic” data of (nice) topological spaces as well as the higher structure of categories, up to equivalence. Topological
and homotopical aspects of simplicial sets are extensively covered in [10] or [3], while their use as a model for \((\infty, 1)\)-categories is covered in [9].

Semi-simplicial sets provide a more “rigid” (thus less powerful) approach to simplicial sets, with the benefit of being easier to define and manipulate. As we are eventually interested in constructing semi-simplicial \textit{types}, different (albeit equivalent) definitions will yield different constructions in the end.

1.4.1 Equivalent definitions

\textbf{Notation.} We write \([n]\) for the nonempty finite set \(\{0, \ldots, n\}\).

\begin{definition}
\textbf{Definition 4.} The semi-simplex category \(\Delta_+\) is defined as having all the sets \([n]\) as objects, and all the \textit{strictly increasing} maps between them as morphisms. A semi-simplicial set is then a presheaf \(X : \Delta_+^{\text{op}} \to \text{Set}\).
\end{definition}

Notice that all the arrows can only go from an object \([n]\) to another \([m]\) with \(n < m\), except for the identity morphisms. \(\Delta_+\) is an example of what is called a \textit{direct category}, and we will soon study them in more details.

The category of semi-simplicial sets is noted \(\text{SSSet}\), and is not to be confused with the category of simplicial sets, similarly noted \(\text{sSet}\). The main difference between the two is that we allow ourselves to look at \textit{nonstrictly} increasing maps as well in the definition of the latter. Now, for the main result:
Theorem 3. The following characterizations of $\text{SSSet}$ are all equivalent:

i) A semi-simplicial set is a presheaf $X : \Delta_+^{op} \longrightarrow \text{Set}$.

ii) A semi-simplicial set is the data of a family $\left( X_n \right)_{n \in \mathbb{N}}$ of sets together with maps $d_i : X_n \longrightarrow X_{n-1}$ such that $0 \leq i \leq n$ called face maps, which satisfy the relation:

$$d_i d_j = d_{j-1} d_i \quad \text{when } i < j$$

iii) Strictly increasing maps $[n] \longrightarrow [m]$ are in bijection with the subsets of $[m]$ of cardinal $n + 1$, thus giving a different definition of $\Delta_+$.

iv) A semi-simplicial set is the data of a discrete right fibration:

$$\Sigma : E \longrightarrow \Delta_+$$

v) $\text{SSSet}$ is the free (small) cocompletion of $\Delta_+$.

Point ii) is actually the standard approach to defining semi-simplicial sets. The category $\Delta_+$ is generated by coface maps $d^i : [n] \longrightarrow [n + 1]$ such that $d^i$ is the only strictly increasing map which “misses” $i$ in its image. Unsurprisingly, the coface maps also satisfy the dual relation:

$$d^i d^j = d^j d^{i-1} \quad \text{when } i < j$$

What is interesting is that it gives us some kind of description of $\Delta_+$ by means of generators and relations, as indeed the following (easy) lemma holds:

Lemma 3. Any strictly increasing map $[n] \longrightarrow [m]$ can be canonically written as a composition of $d^i$’s.

Points i) and iii) are essentially different sets of generators and relations for the category $\Delta_+$. One might then be tempted to say that $\Delta_+$ is “presentable” just as the “presentation” of a group encapsulates the corresponding notion in group theory, but unfortunately the term is already used by category theorists for a distinct (although similar) concept. Instead, a category theorist would rather say that $\Delta_+$ is a “2-polygraph” (or a “2-computad” if they are Australian). I will present the full definition of polygraphs in a later section.

The general idea is this: if we have distinct but equivalent definitions of a category using generators and relations, then from a constructive point of
view we should see distinct but equivalent constructions of presheaves. These distinctions “vanish” in classical mathematics, while they are unavoidable in type theory. The primary goal of this thesis is precisely to better understand these unavoidable distinctions.

Before ending this first section, I will now go back to point ii) and elaborate on the “geometric” aspect of face maps.

1.4.2 Geometric intuition

**Notation.** As $\mathbb{[n]}$ is pretty much unreadable, we write $\Delta^n_+$ for the representable functors of SSSet instead.

**Definition 5.** $\Delta^n_+$ is called the (standard) $n$-(semi-)simplex.

Simplices are “generalized triangles in higher dimensions”. Their direct counterpart in topology are the sets:

\[
\{ (x_1, \ldots, x_n) \in (\mathbb{R}^+)^n \mid x_1 + \cdots + x_n \leq 1 \}
\]

Any map $\Delta^n_+ \to X$ can really be thought as picking a single $n$-simplex from the space $X$. Simplices themselves are made of smaller simplices. If we write $a_0 a_1 \ldots a_n$ the strictly injective map from $[n]$ to some $[m]$ such that:

\[
a_0 < a_1 < \cdots < a_n \in [m]
\]

Then we can easily see that the “filled” triangle $\Delta^2_+$ is made up of three vertices, three segments, and one surface:

\[
(\Delta^2_+)_{0} = \{0, 1, 2\} \\
(\Delta^2_+)_{1} = \{01, 12, 02\} \\
(\Delta^2_+)_{2} = \{012\} \\
(\Delta^2_+)_{3} = \emptyset \\
(\Delta^2_+)_{4} = \emptyset \\
\vdots
\]
The face maps $d_i$ really select the faces of the simplex, as for example:

$$d_2(012) = 01 \quad d_1(12) = 1 \quad d_1(01) = 0$$

i.e. vertices, segments, etc. Semi-simplicial sets are then made of simplices which are glued along common faces indicated by the face maps.

**Remark.** One way of building a 2-sphere is to contract all the edges of a triangle to a point, like a rubber balloon that you would seal at its end.

This is impossible to describe with semi-simplicial sets as the gluing can only occur on faces of matching dimension: we cannot contract a segment to a single point.

Simplicial sets (without the semi-), by contrast, allow for this sort of “compact” constructions at the cost of being even more difficult to use.
2 Grothendieck construction

The Grothendieck construction relates functors $C^{\text{op}} \to U$ into a sufficiently nice category $U$ (e.g. Set or Cat) with “well-behaved” projections. We already have seen an example of it in the previous section, with the correspondence:

$$X : C^{\text{op}} \to \text{Set} \simeq \Sigma : \text{el} X \to C$$

It is essentially equivalent to either look for the list of objects of a given label, or given an object look its label up. If we look at presheaves $X : C^{\text{op}} \to \text{Cat}$ instead, the correspondence in fact still holds — provided the fibers of the projection now additionally has the structure of a category.

2.1 Discrete version

Let $E$ and $C$ be small categories.

**Definition 6.** A discrete right fibration (over $C$) is a functor $\Sigma : E \to C$ such that for any $x \in E$ and any arrow $f : c \to \Sigma(x)$ in $C$, there exists a unique lift $\tilde{f} : y \to x$ in $E$ such that $\Sigma(\tilde{f}) = f$.

If we write $2$ for the category with two objects and a single nontrivial arrow between them and $r : \ast \to 2$ for the canonical inclusion on the right, then the definition of a discrete right fibration can be succinctly formulated with a diagram:

\[
\begin{array}{ccc}
\ast & \to & E \\
\downarrow r & & \downarrow \Sigma \\
2 & \to & C
\end{array}
\]

Remember that Cat is the category of all small categories:

**Notation.** I will use the nonstandard notation $\text{Discr} C$ for the full subcategory of $\text{Cat}/C$ (the slice category over $C$) of discrete right fibrations.

**Theorem 4.** The categories $\text{PSh}(C)$ and $\text{Discr} C$ are equivalent.
The proof is very standard: we define a functor $F : \text{Discr} \mathcal{C} \rightarrow \text{PSh}(\mathcal{C})$, and we show that it is essentially surjective and fully faithful. Let us consider:

$$F : \text{Discr} \mathcal{C} \rightarrow \text{PSh}(\mathcal{C})$$

$$\Sigma : \mathcal{E} \rightarrow \mathcal{C} \rightarrow \Sigma^{-1}$$

It is easy to check that $F$ is well-defined.

On the one hand, given a presheaf $X : \mathcal{C}^{\text{op}} \rightarrow \text{Set}$ we already know that its associated $\Sigma : \text{el} X \rightarrow \mathcal{C}$ is a discrete right fibration. Moreover, we have that:

$$\Sigma^{-1}(c) = \{ (x, c) \mid x \in Xc \} \cong Xc$$

and similarly $\Sigma^{-1}(f) \simeq Xf$ for any arrow $f$ of $\mathcal{C}$. As a result $F(\Sigma) \simeq X$, i.e. $F$ is essentially surjective.

On the other hand, suppose that we have a morphism $R : \mathcal{D} \rightarrow \mathcal{E}$ in $\text{Cat}$ as well as two discrete right fibrations $\Gamma : \mathcal{D} \rightarrow \mathcal{C}$ and $\Sigma : \mathcal{E} \rightarrow \mathcal{C}$. We want to know what necessary and sufficient conditons we need on $R$ to be a morphism in $\text{Discr} \mathcal{C}$, i.e. such that the following diagram commutes:

$$\begin{array}{ccc}
\mathcal{D} & \xrightarrow{R} & \mathcal{E} \\
\downarrow{\Gamma} & & \downarrow{\Sigma} \\
\mathcal{C} & & \\
\end{array}$$

For any $x \in \mathcal{D}$ and $f : c \rightarrow c'$ of $\mathcal{C}$ such that $\Gamma(x) = c'$, there is a unique lift $\tilde{f} : y \rightarrow x$ in $\mathcal{D}$ such that $\Gamma(\tilde{f}) = f$. Similarly, given $R(x) \in \mathcal{E}$ there is a unique lift $\tilde{f} : y' \rightarrow R(x)$ in $\mathcal{E}$ such that $\Sigma(\tilde{f}) = f$.

Clearly, what we need is $R$ to send $\tilde{f}$ to $\tilde{f}$ for any choice of $f$. This is the data of a commutative square:

$$\begin{array}{ccc}
y & \xrightarrow{\Gamma^{-1}(c)} & \xrightarrow{R} \Sigma^{-1}(c) & \xleftarrow{\Sigma^{-1}(f)} & y' \\
\tilde{f} & \downarrow{\Gamma^{-1}(f)} & & \downarrow{\Sigma^{-1}(f)} & \tilde{f} \\
x & \xrightarrow{\Gamma^{-1}(c')} & \xleftarrow{R} \Sigma^{-1}(c') & & R(x) \\
\end{array}$$

which exactly describes a natural transformation between $\Gamma^{-1}$ and $\Sigma^{-1}$, and thus it shows that $F$ is fully faithful. 

$\blacksquare$
2.2 Categorical version

A more general statement of the Grothendieck construction relates “categorical” presheaves with a special sort of fibrations, called Grothendieck fibrations:

\[ X : \mathcal{C}^{\text{op}} \to \text{Cat} \quad \sim \quad \Sigma : \int X \to \mathcal{C} \]

The precise theorem below is adapted from [7, p. 264].

**Definition 7.** The Grothendieck category of a presheaf \( X : \mathcal{C}^{\text{op}} \to \text{Cat} \), noted \( \int X \), is defined as:

- **Objects** are pairs \((c, x)\) for every object \( x \in \mathcal{X}_c\) of the small category \( \mathcal{X}_c\)
- **Morphisms** are pairs \((f, \alpha) : (c, x) \to (c', y)\) of arrows:
  
  \[
  \begin{align*}
  f : c & \to c' \quad \text{of } \mathcal{C} \\
  \alpha : x & \to Xf(y) \quad \text{of } \mathcal{X}_c
  \end{align*}
  \]

  such that, given \((g, \beta) : (c', y) \to (c'', z)\) we have:

  \[
  x \xrightarrow{\alpha} Xf(y) \xrightarrow{Xf(\beta)} Xf(Xg(z)) = X(gf)(z)
  \]

  i.e. composition is defined as \((g, \beta) \circ (f, \alpha) = (gf, Xf(\beta)\alpha)\).

The definition is very similar to the one of categories of elements with presheaves over \(\text{Set}\). Just like then, we have a canonical projection \(\Sigma : \int X \to \mathcal{C}\) defined such that \(\Sigma(c, x) \overset{\text{def}}{=} c\). Even more so, there is an obvious functor:

\[ D : \text{Set} \to \text{Cat} \]

which turns a set \(A\) into the discrete category \(DA\) with \(A\) as its set of objects. Now, if we have \(S : \mathcal{C}^{\text{op}} \to \text{Set}\) we can push it forward into \(\text{Cat}\):

Then indeed we have that \(\text{el } S \equiv \int DS\), i.e. the two notions do coincide.
Remark. In the last step of the definition of composition in $\int X$, we have:

$$Xf(Xg(z)) = X(gf)(z)$$

because $X$ is a “strict” contravariant functor.

One could ask $X : C^{op} \to \text{Cat}$ to be merely a pseudofunctor instead, that is a functor which preserves composition only up to isomorphism. Then the last step would become:

$$Xf(Xg(z)) \xrightarrow{\theta_{fg}} X(gf)(z)$$

where $\theta_{fg}$ is the corresponding isomorphism. Everything that I will present next also applies to pseudofunctors, at the cost of being extra careful with the compositions involved. The fully “pseudofunctorial” approach is the one done in [7].

Again, just like with categories of elements, the projection $\Sigma : \int X \to C$ has special lifting properties. But first, let us have a closer look to the structure of morphisms in $\int X$:

**Definition 8.** A morphism $(f, \alpha) : (c, x) \to (c', y)$ can be decomposed the following way:

- Morphisms with shape $(1_c, \alpha) : (c, x) \to (c, Xf(y))$ lie entirely within the fiber $\Sigma^{-1} \cong Xc$ and are said to be vertical

- Morphisms with shape $(f, 1_{Xf(y)}) : (c, Xf(y)) \to (c', y)$ are said to be horizontal

---

16
The factorization \((f, \alpha) = (f, 1_{Xf(y)}) \circ (1_c, \alpha)\) is clearly unique. Now, for any diagram such as below:

\[
\begin{array}{ccc}
  x & \xrightarrow{\forall \Delta} & y \\
  Xf(y) & \xrightarrow{(f, 1_{Xf(y)})} & y \\
  c & \xrightarrow{f} & c'
\end{array}
\]

where both \(x, Xf(y) \in Xc\), and \(\Sigma(\Delta) = f\). By definition of \(\int X\), there must be a unique vertical arrow \(\phi\) which completes the diagram:

\[
\begin{array}{ccc}
  x & \xrightarrow{\exists! \phi} & y \\
  Xf(y) & \xrightarrow{(f, 1_{Xf(y)})} & y \\
  c & \xrightarrow{f} & c'
\end{array}
\]

In that case \((f, 1_{Xf(y)}) : (c, Xf(y)) \longrightarrow (c', y)\) can be thought as the “canonical lift” of \(f : c \longrightarrow c'\) into \(\int X\) at point \(y\), since any morphism \(\Delta : x \longrightarrow y\) over \(f\) is uniquely an extension of \((f, 1_{Xf(y)})\). This is the special lifting property we are looking for:

**Definition 9.** Let \(\Sigma : \mathcal{E} \longrightarrow \mathcal{C}\) be a functor. A morphism \(\gamma : e \longrightarrow e'\) of \(\mathcal{E}\) is a cartesian lift of a morphism \(f : c \longrightarrow c'\) of \(\mathcal{C}\) if \(\Sigma(\gamma) = f\), and for any other morphism \(\Delta : e'' \longrightarrow e'\) with the same codomain as \(\gamma\) and such that \(\Sigma(\Delta) = f\), there exists a unique \(\phi : e'' \longrightarrow e\) such that \(\Sigma(\phi) = 1_c\) and \(\gamma \phi = \Delta\):
This is exactly the same diagram as before, except for a general category \( E \). Cartesian lifts such as \( \gamma \) are the direct analogue of horizontal morphisms, while the factor \( \phi \) is a vertical morphism. Now, there is no reason for a cartesian lift to be unique a priori — but it is indeed the case:

**Lemma 4.** A cartesian lift, if it exists, is unique up to (unique) isomorphism.

**Proof.** This is a matter of straightforward verification of a universal property.

Given a functor \( \Sigma : E \to C \) and two possible cartesian lifts:

\[
\gamma : e \to e' \quad \text{and} \quad \delta : e'' \to e'
\]

of a morphism \( f : c \to c' \) of \( C \), we must have:

\[
\exists \phi \quad \exists' \phi' \quad \phi \text{ is an isomorphism}
\]

and similarly in the other direction by interverting \( \gamma \) and \( \delta \). This shows that the arrow \( \phi \) in the diagram must be an isomorphism. ■

**Definition 10.** A functor \( \Sigma : E \to C \) is called a Grothendieck fibration if every arrow \( f : c \to c' \) of \( C \) has a cartesian lift.

**Notation.** The full subcategory of \( \text{Cat}/C \) of Grothendieck fibrations is noted \( \text{Fib}_C \).

Grothendieck fibrations are the “correct” notion of fibrations for categorical presheaves. Just as with discrete right fibrations, the lifting property can be succinctly formulated with a diagram:
A nice (and important) property of Grothendieck fibrations is that cartesian lifts are stable under composition, up to isomorphism.

If \( X : C^{\text{op}} \to \text{Cat} \) were a pseudofunctor instead, we would need to relax the definition of a cartesian lift in order to keep track of the isomorphisms. As a result we would also need to make an explicit choice of lifts, and include this additional data in our definition of fibrations — losing the stability under composition in the process. This is done in [7] with the notion of cleavage and cloven fibrations.

Unsurprisingly, the projection \( \Sigma : \int X \to C \) is a Grothendieck fibration. We now know all the ingredients for the main result:

\[
\begin{align*}
\text{Theorem 5.} \quad \text{The categories } [C^{\text{op}}, \text{Cat}] \text{ and } \text{Fib } C \text{ are equivalent.}
\end{align*}
\]

The structure of the proof is essentially the same as in the discrete case. ■

2.3 To infinity and beyond

An even more general statement of the Grothendieck construction exists within the setting of \((\infty, 1)\)-categories, and is covered in [9, p. 168].

For the rest of this thesis, we will focus on the Grothendieck construction in type theory:

\[
X : A \to \text{Type} \simeq \sum_{a : A} X a \to A
\]

which relates dependent types with \(\Sigma\)-types. We will now see how with the definition of presheaves in type theory.
3 Presheaves in type theory

From this point I will adopt a more informal style, as I will only be presenting intuitions for the most part in the absence of concrete results. I assume the reader has a basic knowledge of Martin-Löf type theory (Π-types and Σ-types, identity types, ...) and homotopy type theory (HoTT). Any introduction should suffice, although [11] covers both extensively.

A question raised during the Univalent foundations special year [8] asked whether it would be possible to construct semi-simplicial sets as dependently-typed families of sets, i.e. something along the lines of:

\[ X_0 : \text{Type} \]
\[ X_1 : X_0 \rightarrow X_0 \rightarrow \text{Type} \]
\[ X_2 : \prod_{a \ b \ c : X_0} X_1(a, b) \rightarrow X_1(b, c) \rightarrow X_1(a, c) \rightarrow \text{Type} \]

where \( X_0 \) would be the type of points, \( X_1(a, b) \) would be the type of segments with endpoints \( a \) and \( b \), and so on. One notable issue arising in the setting of HoTT is the need of higher coherence witnesses as the identity types have then unbounded homotopy level. The investigations pertaining to the construction of semi-simplicial types and the higher coherence issues are covered for example in [4] or [1].

3.1 Two possible presentations

There is one “obvious” way of defining semi-simplicial types. We know from our previous section on semi-simplicial sets that we only need the data of a family of sets \((X_n)_{n \in \mathbb{N}}\) and an associated family of face maps \(d_i : X_n \rightarrow X_{n-1}\) which satisfy the relation:

\[ d_i d_j = d_{j-1} d_i \quad \text{when} \ i < j \]

That would informally translate in type theory into a family of types:

\[ X : \text{nat} \rightarrow \text{Type} \]

together with a family of face maps and relations:

\[ d : \prod_{i \ n : \text{nat}} i \leq n \rightarrow X n \rightarrow X (n - 1) \]

\[ \alpha : \prod_{i \ j \ n : \text{nat}} i < j \rightarrow d_i d_j = d_{j-1} d_i \]
It would not be that simple in HoTT as, in general, we need additional data to ensure that the $\alpha$‘s are compatible together higher up in a “coherent” way. One workaround is to truncate, i.e. replace Type with $h\text{Set}$ (that is, types with homotopy level 2).

In any case (truncated or not), we now have two possible approaches to defining semi-simplicial types:

- The *indexed presentation* mentioned earlier with dependent types, where the face maps are mere projections and thus implicit
- The *fibered presentation* I just showed, where the face maps and their relations are explicitly part of the data

Now as both approaches supposedly produce the same semi-simplicial types in the end, there must exist an isomorphism that turns indexed presentations into fibered ones. Consider a much simpler case where, instead of presheaves over $\Delta_+$, we look at presheaves over $2$. An element of $\text{PSh}(2)$ is the data of two sets $X_0, X_1$ together with an application $f : X_1 \rightarrow X_0$. In type theory, this would give:

\[
X_0 : \text{Type} \\
X_1 : X_0 \rightarrow \text{Type}
\]

for the indexed presentation, and:

\[
Y_0, Y_1 : \text{Type} \\
f : Y_1 \rightarrow Y_0
\]

for the fibered one. Mapping $X_0$ to $Y_0$ is direct, but what about $X_1$ and $Y_1$? Clearly, the only sensible choice would be for $Y_1$ to be the “elements” of $X_1$:

\[
Y_1 \overset{\text{def}}{=} \sum_{a : X_0} X_1 a
\]

Then $f$ would be naturally defined as the projection $\pi_1 : \sum X_1 a \rightarrow X_0$. This is exactly the discrete Grothendieck construction!

Of course, we restricted ourselves to the case where the base category is 2 for illustrative purposes. In general, we would have many more dependencies and coherence equations to keep track of in order to build an isomorphism. Even worse, in the case of simplicial sets we have seen that there are multiple ways to describe the base category $\Delta_+$. Making these distinctions more explicit is the subject of the next subsections.
3.2 Reedy categories

Let us be a little bit more rigorous on what we need to give a precise formulation of the indexed and fibered presentations.

It seems unavoidable that in order to turn presheaves $C^{\text{op}} \to \text{Set}$ into a type-theoretic concept, $C$ must have special properties which makes it at least definable in type theory. At the very least, both its objects and hom sets should be definable as types (and, as a consequence, at most countable). Now, in order to build a dependently-typed family of sets, we need an inductive structure on the objects of $C$, i.e. there cannot be any kind of “circular dependencies”. This is where the notion of Reedy category comes in:

**Definition 11.** A Reedy category is a category $C$ such that:

- Every object $c \in C$ has an associated degree $d(c) \in \omega$, where $\omega$ is the first countable ordinal

- There exist two subclasses of arrows of $C$ noted $C_+$ and $C_-$ such that:
  1. Morphisms $c \to c'$ of $C_+$ raise degrees, i.e. $d(c) < d(c')$
  2. Morphisms $c \to c'$ of $C_-$ lower degrees, i.e. $d(c) > d(c')$
  3. Every morphism of $C$ factors uniquely as the composition of a morphism in $C_-$ followed by a morphism in $C_+$

One could also think of $C_+$ and $C_-$ as wide subcategories of $C$.

Many classical examples of finite or countable categories are actually Reedy categories. In particular, the categories $\Delta$ (simplex category) and $\Delta_+$ (semi-simplex category) are Reedy categories. Even more so, $\Delta_+$ really is the wide subcategory of $\Delta$ of degree-raising morphisms!

The fact that every object of a Reedy category has a well-ordered degree opens up the possibility of an indexed presentation of presheaves. Assuming we have the “degree functor” $d : C \to \omega$ and a presheaf $X : C^{\text{op}} \to \text{Set}$, we can decompose $X$ the following way:
where the $X_{d^{-1}(i)}$’s are the presheaf $X$ restricted to the domain $d^{-1}(i)$. Notice how the arrows of $C_+$ and $C_-$ obviously change directions in $C^{op}$.

By extension, I say that the “degree” of a set $Xc$ is the degree of its corresponding object $c$. Now, the arrows of $C_+$ then indicate a dependency of some set $Xd$ on sets $Xc_1, Xc_2, \ldots$ of lower degree. An arrow $f_k : c_k \to d$ of $C_+$ is associated with an application $Xf_k : Xd \to Xc_k$ which, for any $x \in Xd$, tells me that the element $x$ “projects to”, or is “made up of” $f_k(x) \in Xc_k$. From an indexed point of view, this would give in type theory:

$$
\vdots
Xc_1, Xc_2, \ldots
\vdots
Xd : Xc_1 \to Xc_2 \to \cdots \to \text{Type}
$$

Then $x : Xd f_1(x) f_2(x) \cdots$

On the other hand, the arrows of $C_-$ could be thought as “elevating a copy” of an element from a low degree set into a higher degree set. This is much more difficult to express properly in type theory, but an unpublished note by Hugo Herbelin [5] makes a proposition in that direction. As such, I will only focus on categories for which $C_-$ is empty, i.e. what are called direct categories:

**Definition 12.** A direct category is a Reedy category for which there are no degree-lowering morphisms.

Using the same notations as before, if $C$ is a Reedy category then it is a direct category if “$C = C_+$”. $\Delta_+$ is the most immediate example of a direct category, and as a consequence if we have a general theory of presheaves over direct categories in type theory, then in particular we will be able to use it to construct semi-simplicial types.

I have hinted before that the description of a category in terms of generators and relations would play a key role. Let me illustrate this with an elementary example of a direct category that I will call $\mathcal{M}$:
The category $\mathcal{M}$ has three objects $a, b, c$ and four nontrivial arrows $f, g, h, \ell$ such that $gf = \ell = hf$. It is easy to see that $\mathcal{M}$ is indeed a direct category. Now, I could drop $\ell$ in my definition of $\mathcal{M}$ as it is “generated” by $gf$ or $hf$. At the very least, the fibered presentation of a presheaf $X : \mathcal{M}^{\text{op}} \to \text{Set}$ in type theory would change as the generators and relations change:

$$X_a, X_b, X_c : \text{Type}$$

$$X_f : X_b \to X_a$$  

$$X_g : X_c \to X_b$$  

$$X_h : X_c \to X_b$$

Then I either add the data of $X\ell : Xc \to Xa$ and $\alpha : XfXg = X\ell = XfXh$, or I only add the data of $\alpha : XfXg = XfXh$. This is perfectly equivalent, but not strictly equal. From the indexed point of view on the other hand, we would get:

$$X_a : \text{Type}$$

$$X_b : X_a \to \text{Type}$$

$$X_c : \prod_{i : X_a} X_b i \to X_b i \to \text{Type}$$

The relations have been “flattened” into definitional equalities of projections, and the generators have been completely eliminated. Indeed, if we consider:

$$i : X_a$$

$$j, k : X_b i$$

$$x : Xc i j k$$

Then somehow we could say that:

$$g^* x = j \quad h^* x = k \quad \ell^* x = f^* g^* x = f^* h^* x = i$$

provided we interpret $f^*, g^*, h^*, \ell^*$ as being the obvious projections. What happens is that the $\Pi$-type “eats” all of the complexity when we explicitly know how to compute the relations of our category. But, let us say that, instead of a description as explicit as with $\mathcal{M}$ we are given an abstract set of generating
morphisms \(G\), and an abstract set of relations \(R\) identifying some compositions of morphisms of \(G\). As definitional equality is stricter, it would not be possible for us to turn the propositional equalities of \(R\) into definitional ones in general, inside of our \(\Pi\)-type (at least, not this way). In any case, we need to make the notion of generators and relations in the setting of category theory more explicit.

### 3.3 Polygraphs

Given an arbitrary graph, it should be quite easy to imagine how to construct the “free category” over this graph, by formally adding all the finite paths as arrows. To be more precise, the process is the following: take a set of 0-generators \(P_0\) that will serve as objects of our category, and a set of 1-generators \(P_1\) which connect objects to one another, effectively describing arrows. This additionally requires two maps \(s_0, t_0: P_1 \to P_0\) usually called source and target, such that given an arrow \(f \in P_1\) then its “domain” and “codomain” are \(s_0(f)\) and \(t_0(f)\).

It is actually a classical result of abstract nonsense that directed graphs are exactly the data of a presheaf over the category:

\[
\begin{array}{ccc}
\bullet & \xrightarrow{s} & \bullet \\
\downarrow & & \downarrow \\
\bullet & \xrightarrow{t} & \bullet
\end{array}
\]

Now, the pieces I introduced would fit in a (slightly more complete) diagram:

The bottom row contains the structures freely generated by the data on the top row. One could think of \(P_0^*\) as the “0-cells” of the “free 0-category” (i.e. with only objects) generated by \(P_0\). Then we really have \(P_0^* = P_0\), witnessed by the inclusion \(i_0\). We can thus ask for the maps \(s_0, t_0\) to land in \(P_0^*\) instead as it is strictly the same, just like on the diagram. \(P_1^*\) would be, in turn, the “1-cells” of the “free (1-)category” generated by the arrows in \(P_1\) by formally adding all the finite paths as mentioned before, and \(s_0^*, t_0^*\) would be the extensions of \(s_0, t_0\) to \(P_1^*\). Of course, the generating arrows are in particular finite paths, so we have a canonical inclusion \(i_1: P_1 \to P_1^*\) as well. How do we now introduce relations? A relation is merely the choice of two arrows of \(P_1^*\) that we want to be identified. We could further extend our diagram on the right:
The set $P_2$ describes the relations we want on $P_1^*$. More concretely, given an element $r \in P_2$ we ask that $s_1(r) = t_1(r)$, and for that to make sense we need the domains and codomains of the selected arrows to coincide, i.e.:

$$s_0^* s_1 = s_0^* t_1 \quad t_0^* s_1 = t_0^* t_1$$

Then again, $P_2^*$ would be the “2-cells” of the “free 2-category” generated by quotienting $P_1^*$ with the relations in $P_2$. If we keep iterating this process, we get the general definition of a polygraph:

**Definition 13.** An $n$-polygraph is the data of a diagram in $\text{Set}$:

Elements of the $P_k$’s are called $k$-generators. Additionally, the arrows on the diagram should satisfy the following relations:

- The maps $s_k^*, t_k^*$ really are extensions of $s_k, t_k$ through $i_{k+1}$:
  $$s_k^* i_{k+1} = s_k \quad t_k^* i_{k+1} = t_k$$

- Sources and targets should coincide at the previous level:
  $$s_k^* s_{k+1} = s_k^* t_{k+1} \quad t_k^* s_{k+1} = t_k^* t_{k+1}$$

Notice how I did not include $P_n^*$ in the definition. To put it loosely, the idea is that an $n$-polygraph should be the description of a “free $(n-1)$-category” where we already strictly identified some of the “$(n-1)$-cells”, instead of directly
looking at a “free n-category”. A 2-polygraph really is, for example, an ordinary (1-)category which has a set $P_1$ of generators and a set $P_2$ of relations which are “strictly” considered (i.e. the diagrams commute “on the nose”). There are many other higher categorical aspects of polygraphs to consider which are completely out of the scope of this thesis, but which the interested reader can find in the substantial compilation of Ara et al. [2] to be published. In any case, I will really only look at 2-polygraphs for the time being.

Let us go back to our previous example, the category $\mathcal{M}$:

\[
\begin{array}{ccc}
  a & \xrightarrow{f} & b \\
  \quad & \searrow{\ell} & \\
  \quad & \quad & c \\
  \quad & \swarrow{g} & \\
  \quad & \quad & b \\
  \quad & \quad & \quad & \searrow{h} \\
  \quad & \quad & \quad & \quad & c \\
\end{array}
\]

I can now give (at least) two descriptions of $\mathcal{M}$ as a 2-polygraph:

\[
\begin{align*}
P_1 &= \{ f, g, h, \ell \} & P_1 &= \{ f, g, h \} \\
P_2 &= \{ "g f = \ell", "\ell = g f" \} & P_2 &= \{ "g f = h f" \}
\end{align*}
\]

with $P_0 = \{ a, b, c \}$ in both cases. If we go even further back to our main goal of defining presheaves in type theory, then we should have a general procedure that turns 2-polygraphs (and more generally $n$-polygraphs in the “$n$-truncated” case) into either the indexed or the fibered presentation, together with the equivalence between the two. But there is actual room for intermediate steps.

### 3.4 Transitioning from fibered to indexed

Here is one of the possible fibered presentations for the category $\mathcal{M}$, that we already have seen earlier:

\[
\begin{align*}
Xa, Xb, Xc : & \text{ Type} \\
Xf : & \text{ } Xb \rightarrow Xa \\
Xg : & \text{ } Xc \rightarrow Xb \\
Xh : & \text{ } Xc \rightarrow Xb \\
\alpha : & \text{ } XfXg = XfXh
\end{align*}
\]
What we could do is successively “eliminate” the maps using the Grothendieck construction. Let me illustrate this by first eliminating \( Xf \), for example:

\[
\begin{align*}
Xa & : \text{Type} \\
Xb & : Xa \rightarrow \text{Type} \\
Xc & : Xa \rightarrow \text{Type} \\
Xg & : \prod_{i : Xa} Xc i \rightarrow Xb i \\
Xh & : \prod_{i : Xa} Xc i \rightarrow Xb i \\
\alpha & : \pi_1 Xg = \pi_1 Xh
\end{align*}
\]

The maps \( \pi_1 \) are defined as the obvious “mere” projections, through implicit curryification. For example, in the previous case:

\[
\pi_1 : \prod_{i : Xa} Xb i \rightarrow Xa
\]

\[(i, j) \mapsto i\]

Everything that comes “after” \( f \) in the base category \( M \) has to be updated to depend on a choice of term in \( Xa \):

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\]

\[
\begin{array}{c}
\text{f} \\
\text{g} \\
\text{h}
\end{array}
\]

i.e. both \( Xb \) and \( Xc \) now depend on \( Xa \), and the canonical projections correspond to the now-eliminated respective maps \( Xf \) and \( Xf Xg = Xf Xh \). Another possibility would have been to eliminate \( Xg \) first:

\[
\begin{align*}
Xa, Xb & : \text{Type} \\
Xf & : Xb \rightarrow Xa \\
Xc & : Xb \rightarrow \text{Type} \\
Xh & : \prod_{j : Xb} Xc j \rightarrow Xb \\
\alpha & : Xf \pi_1 = Xf Xh
\end{align*}
\]

Again, we only have to update what comes “after” \( g \) in the base category:
Now if I eliminate $Xf$ then $Xg$, I would get:

$$
\begin{align*}
Xa & : \text{Type} \\
Xb & : Xa \rightarrow \text{Type} \\
Xc & : \prod_{i : Xa} Xb \rightarrow \text{Type} \\
Xh & : \prod_{i : Xa} \prod_{j : Xb \rightarrow Xb} Xc \rightarrow Xb \rightarrow \text{Type} \\
\alpha & : \pi_1 \pi_1 \pi_1 = \pi_1 \pi_1 Xh
\end{align*}
$$

This is starting to look a lot like the indexed presentation. If I eliminate $Xh$:

$$
\begin{align*}
Xa & : \text{Type} \\
Xb & : Xa \rightarrow \text{Type} \\
Xc & : \prod_{i : Xa} Xb \rightarrow Xb \rightarrow \text{Type} \\
\alpha & : \pi_1 \pi_1 \pi_1 = \pi_1 \pi_1
\end{align*}
$$

Then indeed, I do get the indexed presentation — up to an extra trivial term $\alpha$.

Overall, it thus seems possible to decompose the equivalence of the fibered and indexed presentations as a succession of “basic” eliminations, corresponding to either direction of the Grothendieck construction. The other direction would be using $\Sigma$-types to turn the mere projections $\pi_1$ back into full-fledged maps. As eliminating maps almost inevitably introduces dependent products (just as above), maybe an easy slogan would be that “fibered and indexed presentations are equivalent, up to replacing $\Sigma$’s with $\Pi$’s and vice versa”. I will now try to make a summary of everything we have seen so far, and outline possible areas of research.

### 3.5 Conclusion

As there is no reason not to take the order of the eliminations into account, we could argue that the possible paths of successive eliminations should be themselves equivalent at a higher level:
Furthermore we only have covered the case of 2-polygraphs here, ignoring the delicate subtleties of fully dealing with the identity type — which would be inevitable in plain HoTT for example. At last, we only really looked at the discrete version of the Grothendieck construction, effectively limiting us in the level of generality we could achieve.

The central intuition is the following: it would be reasonable to expect that “equivalences of equivalences” of possible paths of successive eliminations as in the diagram above, should be again equivalent and higher up. The way these higher equivalences emerge should be reflected in the “higher formulations” of the Grothendieck construction, translated into type theory. In the end, we want a general procedure that turns arbitrary $n$-polygraphs (or “∞-polygraphs” if we can make sense of the notion...?) encoded in type theory in some way and describing a Reedy category, into a “spectrum” — ranging from fibered to indexed — of type-theoretic presentations of presheaves, together with proofs using the Grothendieck construction that these presentations are all equivalent, and proofs that the proofs are all equivalent, and so on.

To be more precise, there are two levels of equivalence:

i) Given a fixed polygraph, all its presentations in type theory (fibered, indexed, and otherwise) should be equivalent

ii) All the polygraphs that describe the same category should be themselves equivalent in some sense

Then, presumably, i) the space of all the possible ways to build presheaves given a fixed polygraph should be contractible and contractibility is exactly the statement of the Grothendieck construction, and ii) the higher category of all the possible ways to build presheaves out of any polygraph describing the same higher category should be equivalent to its category of “higher presheaves”.

30
4 A “Yoneda-esque” presentation

One issue that I have avoided so far is to keep track of the necessary “higher coherence witnesses” I mentioned in the previous introduction. Another aspect I have not touched yet is that, in general, the data of a 2-polygraph is “opaque” in the sense that it is not externally fixed, but a parameter of the procedure we wish to have. Let me illustrate the first point with the semi-simplex category.

Part of this is already explained in the introduction of [4]. Consider the following commutative diagram in $\Delta_+$, assuming $i < j < k$:

\[
\begin{array}{c}
[n+2] & \xrightarrow{d_i} & [n+3] \\
\downarrow & & \downarrow \\
[n+1] & \xrightarrow{d^i} & [n+2] \\
\downarrow & & \downarrow \\
[n] & \xrightarrow{d^i} & [n+1]
\end{array}
\]

The commutation is given by the fact that:

\[d^i d^i = d^i d^{i-1}\] when $x < y$

One can think of this relation as “continuously deforming” the path $d^i d^i$ into the path $d^i d^{i-1}$ and vice versa, on one of the faces of the cube above. Then, still looking at the cube, it seems pretty clear that there are at least two ways to “continuously deform” the path $d^i d^i$ into the path $d^i d^{i-1} d^{k-2}$, either by crossing “at the top” or by crossing “at the bottom” of the cube.

If we tediously named all the faces of the cube, we could formally express this in type theory by showing that there are two distinct proofs of the equality:

\[\pi, \pi' : d^k d^i d^i = d^i d^{i-1} d^{k-2}\]

Then we would need a “higher coherence” proof, witnessing the fact that these two proofs should be equal too (i.e. $\pi = \pi'$) in order for us to define presheaves properly. Of course, this would keep going at higher levels if we draw an hypercube, a “5-cube”, …

One solution seems to render these coherence issues entirely “opaque”, at the cost of introducing new difficulties with dependent typing.
4.1 Staying at Hom

Consider our now-favorite example, the category $\mathcal{M}$:

We have seen that the indexed presentation for $\mathcal{M}$ is the following:

- $X_a : \text{Type}$
- $X_b : X_a \to \text{Type}$
- $X_c : \prod_{i : X_b} X_b i \to X_b i \to \text{Type}$

Notice how the number of dependencies of each type do coincide with the number of ingoing morphisms in the base category, up to equality. For example, on the one hand the only (nontrivial) morphism ending with $b$ is $f : a \to b$, while on the other hand $X_b$ “happens” to only depend on $X_a$. Similarly, the only morphisms ending with $c$ are $g, h, \ell$ and at the same time $X_c$ only depends on two copies of $X_b$ over the same point $i$ (witnessing the fact that $gf = \ell = hf$).

Let us try to factor the dependencies together using the hom “functor” of $\mathcal{M}$:

- $X_a : \text{Type}$
- $X_b : \forall s_0 : \left[ \prod_{f_0 : \text{Hom}(a,b)} X_a \right], \text{Type}$
- $X_c : \forall s_0 : \left[ \prod_{f_0 : \text{Hom}(a,b)} X_a \right], \forall s_1 : \left[ \prod_{f_1 : \text{Hom}(b,c)} X_b \left( \lambda f_0 . s_0 \left( f_1 \circ f_0 \right) \right) \right], \text{Type}$

For the sake of readability, I wrote some of the dependent products with $\forall$ instead of $\Pi$. Let me carefully unpack this:

- As nothing happens at the base level $a$, the type $X_a$ is left unchanged.
- At level $b$, every arrow $f_0 : a \to b$ in the base category should add a new dependency on $X_a$. To choose the dependencies “all at once”, we simply require the data of a dependent function:

$$s_0 : \prod_{f_0 : \text{Hom}(a,b)} X_a$$
In our case, Hom(\(a, b\)) really only contains \(f\) so this is equivalent to choosing a single element in \(X_a\), i.e. just as if we wrote \(X_b : X_a \rightarrow \text{Type}\).

- The last one is the most interesting. We first apply to \(X_c\) exactly the same reasoning to choose the dependencies on \(X_a\):

\[
s_0 : \prod_{f_0 : \text{Hom}(a, c)} X_a
\]

Again, as Hom(\(a, c\)) only contains \(\ell\) this is equivalent to choosing a single element in \(X_a\). But now, \(X_b\) also depends on \(X_a\) so we must ensure that no matter which path we choose, the choices we already made are consistent:

![Diagram](image)

This is why we need to choose the dependencies on \(X_b\) “all at once” the following way:

\[
s_1 : \prod_{f_1 : \text{Hom}(b, c)} X_b (\lambda f_0 \cdot s_0 (f_1 \circ f_0))
\]

If we break down the definition of \(s_1\) we might see how it makes sense, and more importantly how it type checks:

\[
f_0 : \text{Hom}(a, b)
\]
\[
f_1 : \text{Hom}(b, c)
\]
\[
s_0 : \prod_{f_0 : \text{Hom}(a, c)} X_a
\]
\[
s_0 (f_1 \circ f_0) : Xa
\]
\[
\lambda f_0 \cdot s_0 (f_1 \circ f_0) : \prod_{f_0 : \text{Hom}(a, c)} X_a
\]
\[
X_b : \forall s_0 : \left[ \prod_{f_0 : \text{Hom}(a, c)} X_a \right],
\]
\[
\text{Type}
\]

\[
X_b (\lambda f_0 \cdot s_0 (f_1 \circ f_0)) : \text{Type}
\]

Since we always have that \(f_0 = f\) and \(f_1 = g\) or \(f_1 = h\) in the definition above, then by definition of \(\mathcal{M}\) the composition \(f_1 \circ f_0\) will always be equal to \(\ell\), and thus \(X_b (\lambda f_0 \cdot s_0 (f_1 \circ f_0))\) will always be the same “fiber”, i.e. it only depends on \(s_0\). It therefore seems that we have moved the complexity of dealing with the relations of \(\mathcal{M}\) “inside” the Hom!

I will now vastly generalize this to any direct category.
4.2 Definition

Suppose we have the data of a direct category in type theory, encoded as:

\[ A : \text{nat} \to \text{Type} \]
\[ \text{Hom} : \prod_{i, j : \text{nat}} A_i \to A_j \to \text{Type} \]

i.e. \( A \) is a family such that \( A_i \) contains the objects of degree \( i \), and \( \text{Hom}(\_ , \_ ) \) is the underlying hom “functor”. The details of composition, associativity and identity are left out as they are not relevant for now. All the subtleties of generators, relations, and higher coherence witnesses are so far hidden “inside” the computation of \( A_i \) and \( \text{Hom}(x, y) \). To make this more convincing, note that the following easy lemma holds:

**Lemma 5.** A direct category is exactly the data of a category whose objects can be partitioned into a family \((A_i)_{i \in \mathbb{N}}\) such that:

i) All the arrows within any \( A_i \) are trivial, i.e. there are only identities

ii) A nontrivial arrow can only go from \( A_i \) to \( A_j \), with \( i < j \)

As a consequence, if we can formulate an indexed presentation of presheaves only using these ingredients, then presumably we would have successfully overcome the higher coherence issues and made them entirely “opaque”. Here is how we could do it, by extending what we did with the category \( M \) before:

\[ X_0 : A_0 \to \text{Type} \]
\[ X_1 : \forall a_1 : A_1, \]
\[ \forall s_0 : \left[ \prod_{a_0 : A_0} \prod_{f_0 : \text{Hom}(a_0, a_1)} X_0 a_0 \right], \]
\[ \text{Type} \]
\[ X_2 : \forall a_2 : A_2, \]
\[ \forall s_0 : \left[ \prod_{a_0 : A_0} \prod_{f_0 : \text{Hom}(a_0, a_2)} X_0 a_0 \right], \]
\[ \forall s_1 : \left[ \prod_{a_1 : A_1} \prod_{f_1 : \text{Hom}(a_1, a_2)} X_1 a_1 \left( \lambda f_0 . s_0 (f_1 \circ f_0) \right) \right], \]
\[ \text{Type} \]
\[ \vdots \]
\( X_n : \forall a_n : A_n \)

\( \forall s_0 : \left[ \prod_{a_0 : A_0} \prod_{f_0 : \text{Hom}(a_0,a_n)} X_0 a_0 \right], \)

\( \forall s_1 : \left[ \prod_{a_1 : A_1} \prod_{f_1 : \text{Hom}(a_1,a_n)} X_1 a_1 (\lambda f_0 \cdot s_0(f_1 \circ f_0)) \right], \)

\( \cdots \)

\( \forall s_i : \left[ \prod_{a_i : A_i} \prod_{f_i : \text{Hom}(a_i,a_n)} X_i a_i (\lambda f_0 \cdot s_0(f_i \circ f_0)) \cdots (\lambda f_{i-1} \cdot s_{i-1}(f_i \circ f_{i-1})) \right], \)

\( \cdots \)

\( \forall s_{n-1} : \left[ \prod_{a_{n-1} : A_{n-1}} \prod_{f_{n-1} : \text{Hom}(a_{n-1},a_n)} X_{n-1} a_{n-1} (\lambda f_0 \cdot s_0(f_{n-1} \circ f_0)) \cdots \right], \)

**Type**

We now depend on an additional parameter \( A_i \) at each level \( i \), which corresponds to selecting the object of degree \( i \) we are looking at. Everything else is the same as with \( M \), even the intermediate “choices of dependencies”:

\[ \lambda f_k \cdot s_k(f_i \circ f_k) \]

There were no reason *a priori* that its expression would be this simple at any level, but in practice if one were to implement the above definition in a proof assistant such as COQ [14] then one would observe that it indeed works, *provided the associativity of composition holds definitionally!* This last precision is absolutely crucial and I will come back to it in a moment. For now, I would like to explain the name.

### 4.3 Why “Yoneda-esque”?

Let us look at one of the \( s_i \)'s of \( X_n \):

\[ s_i : \prod_{a_i : A_i} \prod_{f_i : \text{Hom}(a_i,a_n)} X_i a_i (\lambda f_0 \cdot s_0(f_i \circ f_0)) \cdots (\lambda f_{i-1} \cdot s_{i-1}(f_i \circ f_{i-1})) \]

Given that \( a_n \) is the very first parameter of \( X_n \), the dependence on \( \text{Hom}(a_i,a_n) \) should strongly remind us of the *representable functor* \( a_n^i \). Then \( f_i \) could be thought as an “\( i \)-edge” of \( a_n^i \), and we would be tempted to write instead:

\[ s_i : \prod_{f_i} (a_n^i) \cdots \]

Let me better illustrate this by taking \( \Delta_+ \) as an example of direct category. If we do the necessary simplifications in the definition, we are left with:
\(X_0 : \text{Type}\)
\(X_1 : \left[ (\Delta_1^1)_0 \longrightarrow X_0 \right] \longrightarrow \text{Type}\)
\(X_2 : \forall s_0 : \left[ (\Delta_2^2)_0 \longrightarrow X_0 \right], \forall s_1 : \left[ \prod_{f_i : (\Delta_2^2)_i} X_1 (\lambda f_0 \cdot s_0(f_1 \circ f_0)) \right], \text{Type}\)
\[
\vdots
\]
\(X_n : \forall s_0 : \left[ (\Delta_n^n)_0 \longrightarrow X_0 \right], \forall s_1 : \left[ \prod_{f_i : (\Delta_n^n)_i} X_i (\lambda f_0 \cdot s_0(f_1 \circ f_0)) \right], \text{Type}\)

The dependency on \(A_i\) has been made implicit, as in the case of \(\Delta^+\) there is only one object of each degree. The geometric intuition should help us understand what happens:

- The base type \(X_0\) contains the points, or vertices.

- \(X_1\) contains the segments. A segment has two endpoints that we can describe with the data of a map \((\Delta_1^1)_0 \longrightarrow X_0\). Indeed \(\Delta_1^+\) is the “standard segment”, so \((\Delta_1^1)_0\) really is the abstract data of two endpoints.

In the end, \(X_1\) returns the type of all segments with given endpoints (i.e. “lying over \((\Delta_1^1)_0 \longrightarrow X_0\)”), hence the signature:

\[X_1 : \left[ (\Delta_1^1)_0 \longrightarrow X_0 \right] \longrightarrow \text{Type}\]

- Now, \(X_2\) contains the “filled” triangles. Just as with \(X_1\), what we do is first select the vertices of a triangle through a map \((\Delta_2^2)_0 \longrightarrow X_0\). Then we need to select three segments between the vertices we have already selected! This what the expression \(\lambda f_0 \cdot s_0(f_1 \circ f_0)\) is for.
If you stare at it long enough, the dependent product:

$$\prod_{f_1 : \Delta^+_2} X_1 \left( \lambda f_0 \cdot s_0(f_1 \circ f_0) \right)$$

guarantees that the selected $f_1$'s have “compatible” endpoints, i.e. one starts where the other ends, and so on. This is what the computation of the composition $f_1 \circ f_0$ does. Overall, given three vertices and three compatible segments $X_2$ returns the type of all the “fillings” of the triangle it describes.

- In the general case, $X_n$ contains the $n$-simplices. Each $s_i$ encodes the data of the “$i$-edges” an $n$-simplex is made of. Because the “$i$-edges” themselves fit inside a “structure” of “$(i - 1)$-edges” and so on, each $s_i$ depends of course on all of the previous ones. This is witnessed by the expressions:

$$\lambda f_k \cdot s_k(f_i \circ f_k)$$

$f_i$ is one of the “$i$-edges” of the standard $n$-simplex. Pre-composing with $f_k$ amounts to looking at one of the “$k$-edges” of $f_i$. By applying $s_k$, we “interpret” the resulting “$k$-edge” $f_i \circ f_k$ in the presheaf we already have constructed.

To sum it all up, an easy slogan would be that “$X_n$ takes the boundary of an $n$-simplex as an argument, and returns the type of all the $n$-simplices with such boundary”.

A different approach would have been to take the indexed presentation for “what it is”, i.e. the way the $n$-simplices interact with each other is directly part of the definition of the presheaf:

$$X_0 : \text{Type}$$
$$X_1 : X_0 \rightarrow X_0 \rightarrow \text{Type}$$
$$X_2 : \prod_{a \ b \ c : X_0} X_1(a, b) \rightarrow X_1(b, c) \rightarrow X_1(a, c) \rightarrow \text{Type}$$

There is no trace of the standard $n$-simplex, or its structure. On the other hand, what we did above is look at the standard $n$-simplex, then “interpret” its structure with the $s_i$’s. How could we not think of the Yoneda lemma:

$$\Delta^n_+ \rightarrow X \cong X_n$$

More exactly, we first looked at the “$i$-edges” of $\Delta^n_+$ so maybe the informal picture to keep in mind is the following:
For example in the most simple case $n = 1$, we can imagine something like:

\[ \Delta^0 \rightarrow X \cong X_n \]

i.e. $X_1$ would be exactly all the ways to fill the arrows $\Delta^0_+ \rightarrow \Delta^1_+$ which are “parallel to” $\Delta^0_+ \rightarrow X$. By applying the Yoneda lemma, we could reformulate this criterion of “parallelism” as the data of an arrow $(\Delta^1_+)_0 \rightarrow X_0$. This is indeed the definition of $X_1$:

\[ X_1 : \left[ (\Delta^1_+)_0 \rightarrow X_0 \right] \rightarrow \text{Type} \]

At this stage, I have three open questions:

i) Is there a way to make this kind of diagram more precise and systematic, for any direct category? In particular, can we visually represent all of the complexity of the “Yoneda-esque” presentation with a diagram?

ii) As types are $\infty$-groupoids in HoTT, can we make “visually explicit” all the choices of paths and compositions that are made along the way?

iii) Suppose we look at “$\infty$-presheaves” instead, from a base $(\infty, 1)$-category to the higher category of $\infty$-groupoids. How can the visual representation help us understand the associativity issues that I will explain next? And how would it relate to a hypothetical “weak $\infty$-Yoneda lemma”?
4.4 Coherence becomes associativity

The “Yoneda-esque” presentation does not type check starting at $X_3$. Let me write down the definition of the first few “slices”:

$$X_0 : A_0 \rightarrow \text{Type}$$

$$X_1 : \forall a_1 : A_1,$$

$$\forall s_0 : \left[ \prod_{a_0 : A_0} \Pi_{f_0} : \text{Hom}(a_0, a_1) \ X_0 a_0 \right] ,$$

$$\text{Type}$$

$$X_2 : \forall a_2 : A_2 ,$$

$$\forall s_0 : \left[ \prod_{a_0 : A_0} \Pi_{f_0} : \text{Hom}(a_0, a_2) \ X_0 a_0 \right] ,$$

$$\forall s_1 : \left[ \prod_{a_1 : A_1} \Pi_{f_1} : \text{Hom}(a_1, a_2) \ X_1 a_1 \ (\lambda f_0 \cdot s_0(f_1 \circ f_0)) \right] ,$$

$$\text{Type}$$

$$X_3 : \forall a_3 : A_3 ,$$

$$\forall s_0 : \left[ \prod_{a_0 : A_0} \Pi_{f_0} : \text{Hom}(a_0, a_3) \ X_0 a_0 \right] ,$$

$$\forall s_1 : \left[ \prod_{a_1 : A_1} \Pi_{f_1} : \text{Hom}(a_1, a_3) \ X_1 a_1 \ (\lambda f_0 \cdot s_0(f_1 \circ f_0)) \right] ,$$

$$\forall s_2 : \left[ \prod_{a_2 : A_2} \Pi_{f_2} : \text{Hom}(a_2, a_3) \ X_2 a_2 \ (\lambda f_0 \cdot s_0(f_2 \circ f_0)) (\lambda f_1 \cdot s_1(f_2 \circ f_1)) \right] ,$$

$$\text{Type}$$

The interesting bit happens in $s_2$:

$$s_2 : \prod_{a_2 : A_2} \Pi_{f_2} : \text{Hom}(a_2, a_3) \ X_2 a_2 \ (\lambda f_0 \cdot s_0(f_2 \circ f_0)) (\lambda f_1 \cdot s_1(f_2 \circ f_1))$$

What is the type of $\phi$ in the expression $\lambda \phi \cdot X_2 a_2 (\lambda f_0 \cdot s_0(f_2 \circ f_0)) \phi$? By doing the necessary substitutions, we get:

$$\phi : \prod_{a_1 \to A_1} \Pi_{f_1} : \text{Hom}(a_1, a_2) \ X_1 a_1 \ (\lambda f_0 \cdot (\lambda f_0 \cdot s_0(f_2 \circ f_0))(f_1 \circ f_0))$$

The composition on the right $\beta$-reduces to $f_2 \circ (f_1 \circ f_0)$, that is:

$$\phi : \prod_{a_1 \to A_1} \Pi_{f_1} : \text{Hom}(a_1, a_2) \ X_1 a_1 \ (\lambda f_0 \cdot (\lambda f_0 \cdot s_0(f_2 \circ f_0))(f_1 \circ f_0))$$

Now, if we look at the type of $\lambda f_1 \cdot s_1(f_2 \circ f_1)$:

$$\lambda f_1 \cdot s_1(f_2 \circ f_1) : \prod_{a_1 \to A_1} \Pi_{f_1} : \text{Hom}(a_1, a_2) \ X_1 a_1 \ (\lambda f_0 \cdot s_0((f_2 \circ f_1) \circ f_0))$$

As a result, we could only substitute $\phi$ with $\lambda f_1 \cdot s_1(f_2 \circ f_1)$ if the following equality held \textit{definitionally}:

$$f_2 \circ (f_1 \circ f_0) = (f_2 \circ f_1) \circ f_0$$

39
This is, of course, not the case in general. A naive workaround would be to try and introduce an additional dependency on proofs of associativity. But then the construction would become quickly unmanageable, as we would also need to depend on proofs of equality of proofs, and so on. This is why it would be interesting to have some sort of diagram of the “Yoneda-esque” presentation that makes the associativity equations more explicit, as I suggested at the end of the previous subsection.

Furthermore, the higher coherence issues we have seen at the beginning (with the “cube of d_i’s”) seem to have been replaced with higher associativity issues. The last question is therefore this: is there an actual, formal correspondence between the higher structure of coherence equations with the higher structure of associativity equations of an \((\infty,1)\)-category? Has “coherence become associativity”?

**Remark.** Another workaround we tried with Hugo Herbelin is to apply the “Yoneda trick” (see [6] for example, called the “Yoneda translation”), which roughly amounts to using the equivalence:

\[
\prod_x \text{Hom}(x,a) \rightarrow \text{Hom}(x,b) \cong \text{Hom}(a,b)
\]

modulo some additional details. In one direction, it sends dependent products \(\gamma\) to \(\gamma_a(1_a)\), and in the other direction it sends morphisms \(g\) to the dependent product \(\lambda x. \lambda f : x \rightarrow a \cdot g \circ f\).

What we supposedly gain with the “Yoneda trick” is that composition is “lifted” to the level of type theory, and thus holds definitionally. Indeed:

\[
h : \prod_x \text{Hom}(x,b) \rightarrow \text{Hom}(x,c)
\]

\[
g : \prod_x \text{Hom}(x,a) \rightarrow \text{Hom}(x,b)
\]

\[
\lambda x. \lambda f : x \rightarrow a \cdot h_x(g_x(f)) : \prod_x \text{Hom}(x,a) \rightarrow \text{Hom}(x,c)
\]

By substituting \(\text{Hom}(a_i,a_n)\) with \(\prod_x \text{Hom}(x,a_i) \rightarrow \text{Hom}(x,a_n)\) in the “Yoneda-esque” presentation, we would presumably overcome the associativity issues. But the equivalence would only hold if every dependent product \(\gamma : \prod_x \text{Hom}(x,a) \rightarrow \text{Hom}(x,b)\) was completely determined by its value at \(\gamma_a(1_a)\)! This is made obvious for any \(f : x \rightarrow a\), by looking at the following diagram:
where $f^*$ is defined as the pre-composition by $f$. This diagram must commute for the equivalence to make sense, and then we conclude that:

$$
\gamma_x(f) = f^*(\gamma_a(1_a))
$$

It is essentially the same idea as in the proof of the Yoneda lemma. But now, our associativity issues seem to have come back as indeed we would probably need this equality at some point:

$$
(f^* \circ (g^* \circ h^*))\gamma_a(1_a) = ((f^* \circ g^*) \circ h^*)\gamma_a(1_a)
$$
References


