



The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

First Year PhD Report

Type Theory through Comprehension Categories

Author:
Paolo CAPRIOTTI

Supervisor:
Venanzio CAPRETTA

July 12, 2013

Contents

Preface	3
1 Models of type theory	5
1.1 Introduction	5
1.2 Locally cartesian closed categories	10
1.3 Grothendieck fibrations	12
1.4 Comprehension categories	17
1.5 Strictification	19
1.6 Other structures	21
1.7 Contextualisation	22
2 Dependent sums and products	25
2.1 Strictification of sums and products	26
3 Intensional equality	28
3.1 Trivial cofibrations	29
3.2 Weak equivalences	30
3.3 Extensional equality	31
4 Universes	34
4.1 Type formers in a restricted model	35

5	Examples	38
5.1	Set model	38
5.2	Groupoid model	38
5.3	Presheaf models	39
5.4	Models in strict ω -categories	42

Preface

This report is an exposition of some of the topics I explored during my first year as a PhD student, and focuses on a purely semantic presentation of type theory and its models.

The main goal of this report is to summarise the most basic results of categorical models of dependent type theory, including an analysis of the issue of strict functoriality of substitution in comprehension categories.

I do not make many claims of originality, but I try to make the presentation as uniform as possible, rely on categorical language and techniques (such as bicategories and profunctors), and take a more conceptual approach, as opposed to the traditional one based on syntax.

The outline is as follows: in chapter 1 I give an informal introduction to the concepts involved in modelling type theory, culminating in the definition of comprehension categories, which are the main object of study in this report.

In chapter 2 I define dependent sums and products for comprehension categories, and state important strictification results (propositions 2.1.2 and 2.1.3).

In chapter 3, intensional identity types are introduced. Following [4], I show how to use identity types to construct a weak factorisation system. Here I apply the original idea of “contextualisation”, introduced in chapter 1, which makes it possible to generalise the result to any split comprehension category.

Chapter 4 is a brief introduction to the idea (due to Vladimir Voevodsky) of using universes to define split comprehension categories out of non-split ones, as an alternative to the strictification construction obtained in proposition 1.5.2.

Finally, chapter 5 is a collection of basic examples that serve as a foundation for richer and more important models like the one on Kan fibrations in [10]. I sketch the construction of the groupoid model [8], and give an original definition of universe for presheaf models.

I also sketch the outline of a new model construction on strict ω -categories which Thorsten Altenkirch and I are currently working on. Developing this model in full, and exploring the possibility of turning it into an implementation of Homotopy Type Theory with the canonicity property, are going to be some of the main objectives of the remainder of my PhD.

Acknowledgements

I am indebted to the members of the Functional Programming Laboratory in Nottingham, for countless hours of very fruitful discussion on type theory and related topics.

In particular, I would like to thank Thorsten Altenkirch, Ambrus Kaposi, Nicolai Kraus, Christian Sattler, and, of course, my supervisor Venanzio Capretta.

Chapter 1

Models of type theory

1.1 Introduction

In this chapter, we will introduce a variety of models of type theory. We will start in the simplest and most direct way, which will naturally lead us to the notion of *locally cartesian closed category*. From there, we will look at possible generalisations, and address the problem of non-strict functoriality of substitution.

To motivate the following definitions, however, we will begin by exploring the basic concepts of intuitive type theory, and show how their desired properties translate directly into categorical structures.

Contexts

The fundamental notion of type theory is that of *dependent type*. For the idea of dependent type to even make sense, however, we first need to state what it is exactly that a type can depend on. This is how we arrive to the notion of *context*.

A context represents a list of assumptions, each assumption being essentially made up of variable name and a type. Every theorem is always stated and proven relatively to some context.

Whenever, in informal mathematics, we say something like “let n be a natural number, R a commutative ring, and M a free R -module of rank n ”, we are effectively defining a context Γ containing the three variables n , R , and M , having the stated types.

This simple example already shows one important characteristic of contexts: the

type of a variable is allowed to depend on previously introduced variables. That is, of course, essential if we want to model the idea of *dependent* types.

Despite the intuition of contexts being essentially lists of pairs, in the following we will take a more axiomatic approach: we will take a collection of contexts \mathcal{C} as given, and work out the structure that this collection ought to possess in order to model the intuitive idea described above.

Morphisms

It is natural to require that contexts form a category.

In fact, assumptions can intuitively be instantiated in the context given by some other assumptions. For example, if Γ denotes the context defined above, with variables n, R, M , and Δ is the context in which we have a natural number m , and field k , we can “interpret” Γ into Δ by setting, for example,

$$\begin{cases} n \mapsto m \\ R \mapsto k \\ M \mapsto k^m \end{cases} \quad (1.1)$$

This would define a morphism from Δ to Γ in the category \mathcal{C} . We will see later, once we have a complete definition of comprehension category, how to make a morphism definitions like (1.1) precise.

If the category \mathcal{C} has a (distinguished) terminal object \bullet , we call \bullet the *empty context*, and think of it as the context where no assumptions have been made. This is consistent with our interpretation, as there should be a unique way to instantiate the empty context in any other context.

Types

Now we can finally move to the central concept: types. Given a context Γ , a type σ over Γ should be defined as something that allows one to talk about:

- the *context extension* $\Gamma.\sigma$, which is to be thought of as the result of adding a new variable of type σ to the existing context Γ
- the *display map* $p_\sigma : \Gamma.\sigma \rightarrow \Gamma$, which is the interpretation of the extended context into the original one obtained by simply “forgetting” about the extra variable.

Note that the above data is exactly what is required to give an object of the slice category \mathcal{C}/Γ . Therefore, any type should determine such an object.

To make this precise in the most general sense, we will need the notion of fibration, introduced in section 1.3. To motivate the general definition, however, we will first leave things at an intuitive level, assume that we have a way to map types over Γ (whatever they are) to objects in \mathcal{C}/Γ , and investigate the structure and properties that this mapping should have.

Terms

Definition 1.1.1. Given a type σ over the context Γ , a **term** M of type σ is a morphism

$$M : \Gamma \rightarrow \Gamma.\sigma$$

that is a section of the display map p_σ , i.e. such that $p_\sigma \circ M = \text{id}$.

The idea of this definition is that a term of type σ is defined to be exactly what is required to give an interpretation of the extended context $\Gamma.\sigma$ in the context Γ . The property of being a section says that the interpretation does not touch any of the other assumptions.

To express the fact that M is term of type σ over the context Γ , we will write the judgement

$$\Gamma \vdash M : \sigma$$

or simply $M : \sigma$, when the context is clear.

Substitutions

Given a morphism $f : \Delta \rightarrow \Gamma$, which we regard as a way to interpret the assumptions in Γ in terms of the assumptions in Δ , there should be a way to transport types and terms over Γ to, respectively, types and terms over Δ . In fact, if the context Γ can be interpreted in Δ , then everything we can state and prove in Γ should make sense in Δ as well.

In particular, given a type σ over Γ , there should exist a type $f^*\sigma$ over Δ , and a morphism $q(f, \sigma) : \Delta.f^*\sigma \rightarrow \Gamma.\sigma$, which we refer to as *f extended with σ* .

The property of being able to transport terms of type σ to terms of type $f^*\sigma$ can be expressed concisely by requiring that the following square

$$\begin{array}{ccc}
 \Delta.f^*\sigma & \xrightarrow{q(f,\sigma)} & \Gamma.\sigma \\
 p_{f^*\sigma} \downarrow & & \downarrow p_\sigma \\
 \Delta & \xrightarrow{f} & \Gamma
 \end{array} \tag{1.2}$$

be a pullback.

In fact, the commutativity property states that the extended morphism behaves like f on the assumptions in Δ , while the universal property of the pullback is equivalent to saying that terms M of type σ can be uniquely transported to terms of type $f^*\sigma$ in a way that is compatible with the extended morphism $q(f, \sigma)$.

If \mathcal{C} has (distinguished) pullbacks, every $f : \Delta \rightarrow \Gamma$ determines a functor $f^* : \mathcal{C}/\Gamma \rightarrow \mathcal{C}/\Delta$, so the condition above can be expressed in any such category. We refer to f^* as the *pullback* (or *substitution*, or *reindexing*) functor.

Dependent products

In order to define a notion of “function” internal to the theory, we need to be able, given types σ and τ over some context Γ , to define a type $\sigma \rightarrow \tau$, whose terms can be thought of as functions from σ to τ .

More generally, given a type σ over Γ , and a type τ over $\Gamma.\sigma$, we want to define a type of *dependent functions* from σ to τ , the so called *dependent product* of σ and τ , which we denote by $\Pi_\sigma\tau$.

Terms of $\Pi_\sigma\tau$ can be thought of as function whose result *type* depends on the argument.

To define dependent products rigorously, we observe that a function of an argument of type σ should be uniquely characterised by the value it assumes on a “generic” variable of type σ . That means that there is a bijective correspondence between terms of type $\Pi_\sigma\tau$ over the context Γ , and terms of type τ over the context $\Gamma.\sigma$.

This condition can be summed up concisely by saying that Π_σ should be a right adjoint of the substitution functor p_σ^* .

Unfortunately, we cannot really make this definition precise, as p_σ^* is a functor between slice categories, while Π_σ should be a mapping of types. We will gloss over this issue for now, and refer to chapter 2 for a precise definition.

The simplified presentation of dependent products given above, however, does motivate the following definition:

Definition 1.1.2. Let \mathcal{C} be a category with pullbacks, and $f : \Delta \rightarrow \Gamma$ a morphism in \mathcal{C} . We say that f is **exponentiable** if the pullback functor $f^* : \mathcal{C}/\Gamma \rightarrow \mathcal{C}/\Delta$ has a right adjoint.

Example 1.1.3. Every morphism is exponentiable in *Set*. This could be proven directly, but it is going to be a simple consequence of proposition 1.2.2 below, so we delay the proof for now.

Dependent sums

The idea of *dependent sum* generalises the notion of binary product.

Given a type σ over Γ , and a type τ over $\Gamma.\sigma$, the dependent sum of σ and τ , denoted $\Sigma_\sigma\tau$, intuitively represents the Γ -type of all pairs of terms M and N , with $M : \sigma$ and $N : M^*\tau$. Categorically, that means that functions from $\Sigma_\sigma\tau$ should be determined by functions from τ in the context $\Gamma.\sigma$.

Therefore, we require Σ_σ to be the left adjoint of the substitution functor $p_\sigma^* : \mathcal{C}/\Gamma \rightarrow \mathcal{C}/\Gamma.\sigma$. This definition is entirely dual to the one of dependent product before.

As for dependent products, we are glossing over the fact that types are, in general, separate from objects of slice categories, and more rigorous presentation will be developed in chapter 2.

Indeed, the simplified notion of dependent sum as presented here is weak enough that it can be defined in any category with pullbacks.

Definition 1.1.4. Let \mathcal{C} be a category with pullbacks. Given a morphism $f : \Delta \rightarrow \Gamma$, we denote by $\Sigma_f : \mathcal{C}/\Delta \rightarrow \mathcal{C}/\Gamma$ the functor given by precomposition with f .

Proposition 1.1.5. Let \mathcal{C} be a category with pullbacks. For any morphism $f : \Delta \rightarrow \Gamma$, the pullback functor $f^* : \mathcal{C}/\Gamma \rightarrow \mathcal{C}/\Delta$ is right adjoint to $\Sigma_f : \mathcal{C}/\Delta \rightarrow \mathcal{C}/\Gamma$.

Proof. For $g : A \rightarrow \Delta$ and $h : B \rightarrow \Gamma$, we need to find an isomorphism

$$\mathcal{C}/\Gamma(f \circ g, h) \cong \mathcal{C}/\Delta(g, f^*h), \quad (1.3)$$

natural in f and g .

The left hand side is the set of all morphisms $A \rightarrow B$ that make the following diagram commute:

$$\begin{array}{ccc} A & \longrightarrow & B \\ g \downarrow & & \downarrow h \\ \Delta & \xrightarrow{f} & \Gamma \end{array}$$

By the universal property of the pullback of h along f , each such morphism uniquely determines a morphism $A \rightarrow B \times_{\Gamma} \Delta$ over Δ , i.e. an element of the set on the right hand side of (1.3).

This defines the isomorphism in (1.3), whose naturality is then easy to verify. \square

The following lemma will be useful later.

Lemma 1.1.6. *Let \mathcal{C} be a category with pullbacks, and $f : \Delta \rightarrow \Gamma$ any morphism. The functor Σ_f is comonadic.*

Proof. The functor Σ_f has a right adjoint by proposition 1.1.5. Furthermore Σ_f clearly reflects isomorphisms, and it is not hard to see that it preserves equalisers. Therefore, Σ_f is comonadic by the dual of Beck's monadicity theorem (see for example [2]). \square

1.2 Locally cartesian closed categories

The most direct way to formalise the ideas presented in section 1.1 is to define types directly as objects of a slice category. This approach was first explored by Seely ([14]).

The biggest advantage of this approach is that the simplified definitions of dependent sums and products given above do actually make sense. Since we are mostly interested in modelling a type theory that does have sums and products, it is natural to restrict ourselves to categories for which the dependent product functor is defined.

Definition 1.2.1. A **locally cartesian closed category** is a category with pullbacks where every morphism is exponentiable.

The choice of terminology is motivated by the following characterisation:

Proposition 1.2.2. *Let \mathcal{C} be a category with pullbacks. The following conditions are equivalent:*

- \mathcal{C} is locally cartesian closed
- all slice categories of \mathcal{C} are cartesian closed

Proof. Clearly all slices are cartesian, since products in a slice category are pullbacks in \mathcal{C} .

Let Γ be an object of \mathcal{C} , and $f : \Delta \rightarrow \Gamma$ an object in \mathcal{C}/Γ . The fibred product functor $- \times_{\Gamma} \Delta$ can be factored as:

$$\mathcal{C}/\Gamma \xrightarrow{f^*} \mathcal{C}/\Delta \xrightarrow{\Sigma_f} \mathcal{C}/\Gamma$$

Slice categories of \mathcal{C}/Γ are cartesian closed if and only if all such fibred product functors have right adjoints.

Since Σ_f is comonadic by Lemma 1.1.6, the adjoint lifting theorem (see [2]) implies that this condition is equivalent to f^* having a right adjoint, that is, to \mathcal{C} being locally cartesian closed. \square

As promised, we can now prove the result stated in example 1.1.3.

Proposition 1.2.3. *The category \mathbf{Set} is locally cartesian closed.*

Proof. For any set X , the category \mathbf{Set}/X is isomorphic to the presheaf category $[X, \mathbf{Set}]$ where X is regarded as a discrete category.

Since presheaf categories are cartesian closed, \mathbf{Set} is locally cartesian closed by proposition 1.2.2. \square

Substitution in locally cartesian closed categories

If \mathcal{C} is a locally cartesian closed category, then all the constructions in section 1.1 are well defined. So in the following we will refer to objects of \mathcal{C} as contexts, objects in \mathcal{C}/Γ as types over Γ , and sections of types as terms.

Given contexts Γ and Δ in \mathcal{C} , and a type σ over Γ , let us examine the mapping given by $f \mapsto f^*\sigma$, for morphisms $f : \Delta \rightarrow \Gamma$. Clearly, this mapping is functorial modulo unique isomorphisms, i.e. given $f : \Delta \rightarrow \Gamma$, and $g : \Theta \rightarrow \Delta$,

$$g^*(f^*\sigma) \cong (f \circ g)^*\sigma$$

Of course, we cannot expect the above isomorphism to be an identity in general. We say therefore that pullbacks are not strictly functorial. Indeed, in most examples of locally cartesian closed categories (including Set), there is no choice of pullbacks for which this property holds.

Strict functoriality is, however, a desirable property for a model of type theory. In fact, if we think of types as some sort of *syntactic* construct, i.e. built inductively out of a few basic primitives, pullbacks corresponds to *syntactic substitutions*, which are clearly strictly functorial.

To better reflect the properties of such syntactic constructs¹, therefore, we need to expand the class of categories that we are interested in, as to include structures for which the corresponding notion of substitution is strictly functorial.

The basic idea, expanded and made precise in the following sections, is to separate the notions of type and object of a slice category.

1.3 Grothendieck fibrations

In Set , it is easy to see that, given an object X , families indexed over X are the same as functions to X . We already made use of this fact in the proof of proposition 1.2.3.

For categories, this doesn't work equally well. In particular, given a functor $p : \mathcal{E} \rightarrow \mathcal{C}$, although we can still define a function $\mathcal{C} \rightarrow \text{Cat}$ that assigns to every object $x : \mathcal{C}$, the *fibre* of p over x , there doesn't seem to be any way to extend it to a functor.

The aim of this section is to define a subclass of functors, called *fibrations*, such that the correspondence between slices and presheaves that we have for Set can be generalised to an equivalence of 2-categories:

$$\text{Fibrations over } \mathcal{C} \cong \text{Pseudofunctors } \mathcal{C}^{\text{op}} \rightarrow \text{Cat} \quad (1.4)$$

To motivate the definition of fibration, we begin by studying the family of slice categories of a category with pullbacks \mathcal{C} , for which, intuitively, the corresponding fibration is going to be the codomain functor $\text{cod} : \mathcal{C}^I \rightarrow I$.

Morphisms in \mathcal{C}^I are commutative squares in \mathcal{C} . Can we express the property of being a pullback square solely in terms of the functor cod ? Consider the following general definition:

¹We will make this notion precise when defining names and substitution in section 1.4

Definition 1.3.1. Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be a functor, and $\beta : y' \rightarrow y$ a morphism in \mathcal{E} . We say that β is **cartesian** if for every morphism $\beta'' : y'' \rightarrow y$ in \mathcal{E} , and every morphism $\alpha' : p y'' \rightarrow p y'$ in \mathcal{C} such that $p \beta'' = \alpha' \circ \alpha'$, there exists a unique $\beta' : y'' \rightarrow y$ in \mathcal{E} such that $p \beta' = \alpha'$.

It is not hard to see that a commutative square in \mathcal{C} is a pullback if and only if the corresponding morphism in \mathcal{C}^I is cartesian.

Now, existence of pullbacks (which, intuitively, should correspond to cod being a fibration), can be expressed easily in terms of cod and the previous definition.

In general, we can say:

Definition 1.3.2. A **(Grothendieck) fibration** is a functor $p : \mathcal{E} \rightarrow \mathcal{C}$, with the property that, for every $y : \mathcal{E}$, and every morphism $\alpha : x' \rightarrow x$ in \mathcal{C} , with $p y = x$, there is a cartesian morphism $\beta : y' \rightarrow y$ such that $p \beta = \alpha$.

In definition 1.3.2, any β such that $p \beta = \alpha$ is called a *cartesian lifting* for α .

Definition 1.3.3. Let $p : \mathcal{E} \rightarrow \mathcal{C}$ and $p' : \mathcal{E}' \rightarrow \mathcal{C}$ be fibrations. A functor $F : \mathcal{E} \rightarrow \mathcal{E}'$ is called **cartesian** if the following diagram

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{F} & \mathcal{E}' \\ & \searrow p & \swarrow p' \\ & \mathcal{C} & \end{array}$$

commutes, and F maps cartesian morphisms to cartesian morphisms.

Fibrations $\mathcal{E} \rightarrow \mathcal{C}$ over a fixed small category \mathcal{C} , with \mathcal{E} small, form a subbicategory of the slice category Cat/\mathcal{C} , where 1-cells are cartesian morphisms. We denote this bicategory by $\text{Fib}_{\mathcal{C}}$.

Proposition 1.3.4. *The codomain functor $\text{cod} : \mathcal{C}^I \rightarrow \mathcal{C}$ is a fibration if and only if \mathcal{C} has pullbacks.*

Proof. Direct consequence of the definitions. □

A choice of cartesian liftings for a fibration is called a *cleavage*. A fibration equipped with a cleavage is called a *cloven fibration*. For example, a cleavage for the codomain fibration is a choice of pullbacks.

In the following, we will assume that all fibrations are cloven.

Given a fibration $p : \mathcal{E} \rightarrow \mathcal{C}$, an object $y : \mathcal{E}$, and a morphism $\alpha : x' \rightarrow x$, with $py = x$, we will denote the selected cartesian lifting of α (from y) with $\bar{\alpha} y : \alpha^* y \rightarrow y$.

Definition 1.3.5. Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be any functor, and $x : \mathcal{C}$. The **fibre** of \mathcal{E} over x (along p) is the subcategory of \mathcal{E} consisting of all those objects that are mapped to x , and all those morphisms that are mapped to the identity of x .

When the functor p is clear, we will denote the fibre of \mathcal{E} over x with \mathcal{E}_x . Morphisms in \mathcal{E} which reside in a fibre are sometimes called *vertical*.

Proposition 1.3.6. Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be a fibration, and $\alpha : x' \rightarrow x$ a morphism in \mathcal{C} . Let j_x be the inclusion $\mathcal{E}_x \rightarrow \mathcal{E}$ (and similarly $j_{x'}$).

The function determined by α^* can be uniquely extended to a functor $\alpha^* : \mathcal{E}_x \rightarrow \mathcal{E}_{x'}$ in such a way that $\bar{\alpha}$ determines a natural transformation $\bar{\alpha} : j_{x'} \circ \alpha^* \rightarrow j_x$

We will refer to the functor α^* as the *substitution* (or *reindexing*) functor determined by α .

Definition 1.3.7. We say that a functor $p : \mathcal{E} \rightarrow \mathcal{C}$ is **small** if for all $x : \mathcal{C}$, the fibre \mathcal{E}_x is a small category.

Now we are ready to make (1.4) explicit. Given a small fibration $p : \mathcal{E} \rightarrow \mathcal{C}$, define a mapping $dp : \mathcal{C}^{\text{op}} \rightarrow \text{Cat}$ by:

$$\begin{aligned} H x &= \mathcal{E}_x \\ H \alpha &= \alpha^* \end{aligned} \tag{1.5}$$

Proposition 1.3.8. The mapping dp defined by (1.5) is a pseudofunctor.

Following section 1.2, we are interested in strictly functorial substitutions. In the language of fibrations, this translates to the following definitions:

Definition 1.3.9. A small fibration $p : \mathcal{E} \rightarrow \mathcal{C}$ is called *split* if the pseudofunctor dp is strict.

It is sometimes useful to consider even stronger conditions on fibrations.

Definition 1.3.10. A small fibration $p : \mathcal{E} \rightarrow \mathcal{C}$ is called *discrete* if dp factors through the functor $\delta : \text{Set} \rightarrow \text{Cat}$ which maps every set X to the discrete category on X .

The bicategory of profunctors

We know that every small fibration determines a pseudofunctor $\mathcal{C}^{\text{op}} \rightarrow \text{Cat}$, but what about small functors that are not fibrations?

Since we can take fibres of arbitrary functors, we can still define an object-level function $\mathcal{C} \rightarrow \text{Cat}$, but it is not at all clear how to extend it to morphisms.

The idea that makes it possible to complete this construction is to enlarge Cat by weakening the notion of functor:

Definition 1.3.11. Let \mathcal{A} and \mathcal{B} be categories. A profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ is simply a functor $\mathcal{B}^{\text{op}} \times \mathcal{A} \rightarrow \text{Set}$.

Intuitively, profunctors are to functors as relations are to functions.

In fact, in the same way as every function determines a relation, every functor determines a profunctor: given $F : \mathcal{A} \rightarrow \mathcal{B}$, we define the profunctor $F^\sharp : \mathcal{A} \nrightarrow \mathcal{B}$ (called *representable*) by

$$F^\sharp(b, a) = \mathcal{B}(b, F a). \quad (1.6)$$

In particular, the identity functor corresponds to the profunctor given by homsets.

Profunctors can be composed. Given $F : \mathcal{A} \nrightarrow \mathcal{B}$, $G : \mathcal{B} \nrightarrow \mathcal{C}$, their composition is defined by the coend:

$$G \circ F = \int^{b:\mathcal{B}} G(-, b) \times F(b, -) \quad (1.7)$$

Note that if \mathcal{B} is small, the coend (1.7) always exists.

Definition 1.3.12. The bicategory Prof of profunctors is defined as follows. The 0-cells of Prof are small categories, the 1-cells are profunctors, and the 2-cells are natural transformations.

Natural transformations of profunctors are defined by regarding a profunctor $\mathcal{B} \rightarrow \mathcal{A}$ directly as a functor $\mathcal{B}^{\text{op}} \times \mathcal{A} \rightarrow \text{Set}$. Composition and identity of 2-cells are as usual.

Composition and identity of 1-cells are defined as above. Associativity and identity isomorphisms can be obtained via coend calculus.

There is an obvious faithful pseudofunctor $\text{Cat} \rightarrow \text{Prof}$, which is the identity on objects, and maps every functor to the corresponding pseudofunctor as in (1.6). We will therefore consider Cat as a subcategory of Prof .

Now, given a small functor $p : \mathcal{E} \rightarrow \mathcal{C}$, and a morphism $\alpha : x' \rightarrow x$ in \mathcal{C} , we can define a profunctor $\alpha^\sharp : \mathcal{E}_x \dashv \mathcal{E}_{x'}$ as follows:

$$\alpha^\sharp(y', y) = \{f : y' \rightarrow y \mid p f = \alpha\}$$

We can then generalise proposition 1.3.8 as follows:

Proposition 1.3.13. *Any small functor $p : \mathcal{E} \rightarrow \mathcal{C}$ determines a normal lax functor $dp : \mathcal{C} \rightarrow \mathbf{Prof}$ given by:*

$$\begin{aligned} H x &= \mathcal{E}_x \\ H \alpha &= \alpha^\sharp \end{aligned} \tag{1.8}$$

Proposition 1.3.14. *A small functor $p : \mathcal{E} \rightarrow \mathcal{C}$ is a fibration if and only if dp is a \mathbf{Cat} -valued pseudofunctor.*

Proof. If p is a fibration, then the general definition of dp for functors (proposition 1.3.13) coincides with the one for fibrations (proposition 1.3.8), thus dp is \mathbf{Cat} -valued and pseudo.

Conversely, suppose dp factors through \mathbf{Cat} . This means that α^\sharp is a representable profunctor for all morphisms $\alpha : x' \rightarrow x$ in \mathcal{C} . Hence there exists a functor $\alpha^* : \mathcal{E}_x \rightarrow \mathcal{E}_{x'}$, and a natural isomorphism:

$$\{\beta : \mathcal{E}(y', y) \mid p \beta = \alpha\} \cong \mathcal{E}_{x'}(y', \alpha^* y) \tag{1.9}$$

for all $y : \mathcal{E}_x$ and $y' : \mathcal{E}_{x'}$.

In particular, choosing $y' = \alpha^* y$, we get a morphism $\bar{\alpha} y : \mathcal{E}(\alpha^* y, y)$ corresponding to the identity through the isomorphism (1.9).

It remains to show that $\bar{\alpha} y$ is cartesian. Suppose then that $\alpha' : x'' \rightarrow x'$ is a morphism in \mathcal{C} , and let $\beta'' : y'' \rightarrow y$ be a lifting of $\alpha \circ \alpha'$. The fact that dp is a pseudofunctor implies that the canonical map

$$(\alpha')^\sharp(y'', \alpha^* y) \cong (\alpha \circ \alpha')^\sharp(y'', y) \rightarrow (\alpha \circ \alpha')^\sharp(y'', y),$$

is an isomorphism. This map is clearly given by composition with $\bar{\alpha} y$, and the property that it is an isomorphism just expresses the fact that $\bar{\alpha} y$ is cartesian. \square

The Grothendieck construction

In section 1.3 we showed how to construct a normal lax functor $\mathcal{C} \rightarrow \mathbf{Prof}$ given any small functor $\mathcal{E} \rightarrow \mathcal{C}$. In this section, we are going to detail the opposite construction.

Let's suppose that $H : \mathcal{C} \rightarrow \mathbf{Prof}$ is a normal lax functor. Define \mathcal{E} as the class of all pairs (x, y) , where $x : \mathcal{C}$, and $y : H x$.

We can build a category structure on \mathcal{E} by defining $\mathcal{E}((x', y'), (x, y))$ to be the set of all pairs (α, β) , where $\alpha : x \rightarrow x'$ is a morphism in \mathcal{C} , and $\beta : H \alpha(y', y)$.

Since $H \text{id} = \text{id}$, we define the identity morphism in $\mathcal{E}((x, y), (x, y))$ to be just the identity morphism in $H \text{id}(y, y) = H x(y, y)$.

The composition of morphisms (α, β) and (α', β') is given by $(\alpha \circ \alpha', H(\alpha, \alpha')[\beta', \beta])$, where $H(\alpha, \alpha') : H \alpha' \circ H \alpha \rightarrow H(\alpha \circ \alpha')$, and $[\beta', \beta]$ is the element in the coend $H \alpha' \circ H \alpha$ determined by the pair (β', β) .

Furthermore, there is an obvious small functor $\mathcal{E} \rightarrow \mathcal{C}$. We will denote the category \mathcal{E} by $\int H$.

These two constructions are inverses of one another. To make this statement precise, we first need a definition.

Definition 1.3.15. Let \mathcal{A} be a bicategory, and $F, G : \mathcal{A} \rightarrow \mathbf{Prof}$ be lax functors. A lax natural transformation $\alpha : F \rightarrow G$ is said to be **representable** if for all $a : \mathcal{A}$, its component $\alpha_a : F a \rightarrow G a$ is a representable profunctor.

Now we can prove:

Proposition 1.3.16 (Bénabou). *The mappings $p \mapsto \text{dp}$ and $H \mapsto \int H$ define a biequivalence between the slice 2-category \mathbf{Cat}/\mathcal{C} , and the bicategory of normal lax functors, representable lax natural transformations, and modifications.*

To conclude the section, we state a classical result, which will be used in section 5.2 to define a fibration over the category of groupoids.

Theorem 1.3.17 (Giraud [5], Conduché [3]). *Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be a functor between small categories. The following are equivalent:*

- (i) p is exponentiable in \mathbf{Cat}
- (ii) dp is a pseudofunctor

1.4 Comprehension categories

The machinery of fibrations allows us to define a very general notion of model of type theory. In the following, \mathcal{C} will denote any category, which we think of as the category of contexts.

Definition 1.4.1. A **comprehension category** is a functor $\text{ext} : \mathcal{E} \rightarrow \mathcal{C}^I$ (where \mathcal{C}^I denotes the *arrow category* of \mathcal{C}), such that:

- (i) the composition $\text{dom} \circ \text{ext}$ is a small fibration
- (ii) ext maps cartesian morphisms to pullback squares

The functor ext is called *context extension functor*.

A comprehension category is called **split** if the fibration $\text{dom} \circ \text{ext}$ is split, and it is called **full** if the functor ext is fully faithful.

Given a comprehension category over \mathcal{C} , we can adopt the terminology and notation of section 1.1, and talk about types, display maps, terms, and substitutions.

In particular, if σ is a type over Γ , and $f : \Delta \rightarrow \Gamma$ a morphism in \mathcal{C} , we have the pullback square (1.2).

Clearly, any category with pullbacks admits a canonical comprehension category structure, by taking ext to be the identity.

Notational conventions

Every morphism $g : \Delta \rightarrow \Gamma.\sigma$ is uniquely determined by a morphism $f : \Delta \rightarrow \Gamma$ and a term $M : f^*\sigma$ in the context Δ . In this case, we write $g = \langle f, M \rangle$, and we say that g is obtained by extending f with the term M .

Extension of morphisms can be iterated. Given a context Γ' of the form $\Gamma.\sigma_1 \dots \sigma_k$, we can uniquely define a morphism $\Delta \rightarrow \Gamma'$ by giving a morphism $f : \Delta \rightarrow \Gamma$, and a sequence M_1, \dots, M_k of terms in the context Δ of the appropriate types. We simply write $\langle f, M_1, \dots, M_n \rangle$ to denote the result of the iterated extension.

Definition 1.4.2. Given a type σ over the context Γ , the **variable** of type σ is the unique term v_σ of type $p_\sigma^*\sigma$ such that $p_{p_\sigma^*\sigma} \circ v_\sigma = \text{id}$.

Definition 1.4.3. A **weakening morphism** is inductively defined as follows:

- a display map is a weakening morphism
- if f is a weakening morphism and σ is a type, $q(f, \sigma)$ is a weakening morphism.

Substituting along weakening morphisms is called *weakening*.

Provided the context is explicit, we will omit substitutions along weakening morphisms when writing down types and terms, as it can usually be inferred. For example, we can say that v_σ is a term of type σ in the context $\Gamma.\sigma$, leaving the substitution p_σ^* implicit, but making the context clear.

As an additional notational convenience, it is often useful to give names to variables. We will do so by defining contexts using a notations like in the following example:

$$\llbracket \Gamma(x : \sigma)(y : \tau) \rrbracket \tag{1.10}$$

This defines the same context as simply $\Gamma.\sigma.\tau$, but fixes the names x and y to refer to the corresponding variables (and weakenings thereof).

When defining a morphism by iterated extension into a context with named variables, we will adopt the notation exemplified by (1.1). For example, if σ and τ are types over Θ , $\Gamma = \llbracket \Theta(x : \sigma)(y : \tau) \rrbracket$, and $\Delta = \llbracket \Theta(u : \tau)(v : \sigma)(z : \tau) \rrbracket$, the following mapping:

$$\begin{cases} u \mapsto y \\ v \mapsto x \\ z \mapsto y \end{cases}$$

defines a morphism $\Gamma \rightarrow \Delta$.

Finally, when working with *split* comprehension categories, it is often useful to make the dependency of types and terms on variables explicit. To do so, we can choose to denote a type $\bar{\rho}$ in a context with named variables x_1, \dots, x_k as a function ρ of k arguments.

The function ρ is defined as follows: given terms of the appropriate types M_1, \dots, M_k , $\rho(M_1, \dots, M_k)$ is the type $\langle \text{id}, M_1, \dots, M_k \rangle^* \bar{\rho}$. In particular, $\bar{\rho}$ itself can be expressed as $\rho(x_1, \dots, x_k)$. The same applies to terms.

1.5 Strictification

Split comprehension categories are more desirable than their non-split counterparts, but are that much harder to find in practice.

Categories commonly considered in mathematics, like algebraic categories, or subcategories of Cat , do not usually have strictly functorial pullbacks, thus their canonical comprehension category structure is not split.

However, we will show that it is always possible to modify any fibration in such a way that it becomes split, and, as a consequence, any comprehension category can be made split by slightly “refining” its notion of type.

Lemma 1.5.1. *Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be a fibration between small categories, and $\Gamma : \mathcal{C}$. There is an equivalence of categories:*

$$\mathcal{E}_\Gamma \cong \text{Fib}_{\mathcal{C}}(\text{dom}_\Gamma, p) \quad (1.11)$$

where $\text{dom}_\Gamma : \mathcal{C}/\Gamma \rightarrow \mathcal{C}$ is the domain fibration.

Proof. Given $E : \mathcal{E}_\Gamma$, the assignment $f \mapsto f^*E$ can be extended to a functor $F E : \mathcal{C}/\Gamma \rightarrow \mathcal{E}$, thanks to the universal property of cartesian liftings. By construction, we have a commutative triangle:

$$\begin{array}{ccc} \mathcal{C}/\Gamma & \xrightarrow{F E} & \mathcal{E} \\ & \searrow & \swarrow p \\ & \mathcal{C} & \end{array}$$

Given $h : E \rightarrow E'$ in \mathcal{E}_Γ , and $f : \mathcal{C}/\Gamma$, we have a commutative diagram:

$$\begin{array}{ccc} f^*E & \longrightarrow & E \\ \downarrow & & \downarrow \\ f^*E' & \longrightarrow & E' \end{array}$$

where the left arrow is again given by the universal property of cartesian liftings. This determines a natural transformation $F h : F E \rightarrow F E'$. By the uniqueness part of the universality, F is a functor $\mathcal{E}_\Gamma \rightarrow \text{Cat}/\mathcal{C}(\text{dom}_\Gamma, p)$.

To show that F is full, fix $E E' : \mathcal{E}_\Gamma$, and let $\eta : F E \rightarrow F E'$ be any natural transformation. The component of η at the identity of Γ is a morphism $\text{id}^*E \rightarrow \text{id}^*E'$. Since id^*X is canonically isomorphic to X for any $X : \mathcal{E}$, we obtain a morphism $h : E \rightarrow E'$, and it's easy to verify that $F h$ is indeed η .

To show that F is faithful, consider morphisms $h k : E \rightarrow E'$, and assume that $F h = F k$. Then in particular their components on the identity of Γ are equal, and it immediately follows that $h = k$.

Finally, we need to show that F is essentially surjective. Let $G : \mathcal{C}/\Gamma \rightarrow \mathcal{E}$ be any functor over \mathcal{C} , and set $E = G \text{id}_\Gamma$. We will exhibit a natural isomorphism η

between G and $F E$. For $f : \mathcal{C}/\Gamma$, we have a commutative triangle:

$$\begin{array}{ccc} \Delta & \xrightarrow{f} & \Gamma \\ & \searrow f & \swarrow = \\ & & \Gamma \end{array}$$

So we get a cartesian morphism $Gf \rightarrow E$ over f . We can now define $\eta_f : Gf \rightarrow f^*E$ using the universal property of the cartesian lifting $f^*E \rightarrow E$. The naturality of η , and the fact that it is an isomorphism, follow easily. \square

Now we are ready to prove:

Proposition 1.5.2. *Let $p : \mathcal{E} \rightarrow \mathcal{C}$ be fibration between small categories. Then there is a split fibration which is equivalent to \mathcal{E} over \mathcal{C} .*

Proof. The assignment $\Gamma \mapsto \text{Fib}_{\mathcal{C}}(\text{dom}_{\Gamma}, p)$ defines a strict 2-functor $H : \mathcal{C} \rightarrow \text{Cat}$. The equivalence of lemma 1.5.1 then gives a cartesian equivalence of fibrations between the fibration $\int H$ and p . \square

Applying proposition 1.5.2 to comprehension categories we get:

Corollary 1.5.3. *Let \mathcal{C} be a comprehension category. Then there is an equivalent split comprehension category structure on \mathcal{C} .*

We call the split comprehension category obtained in this way the *strictification* of \mathcal{C} .

By unfolding the construction, the strictification of \mathcal{C} is essentially obtained by defining types over \mathcal{C} to be a pair of an original type, together with a “coherent” way to substitute this type along any morphism.

1.6 Other structures

In the existing literature about models of type theory, a number of different categorical structures have been considered. Fortunately, they can all be regarded as special cases of comprehension categories.

Definition 1.6.1. A **category with attributes** is a full split comprehension category.

Since the functor $\text{ext} : \mathcal{E} \rightarrow \mathcal{C}^I$ in a category with attributes is fully faithful, we can recover the category structure on \mathcal{E} by just knowing the set of objects over any context.

In other words, a category with attributes can be simply described by giving a functor $\text{Ty} : \mathcal{C} \rightarrow \text{Set}$, and a functor $\text{ext} : \int \mathcal{C} \rightarrow \mathcal{C}^I$, which is the way it is usually defined.

Definition 1.6.2. A **display map category** is a comprehension category where the functor ext is the inclusion of a full subcategory of \mathcal{C}^I .

For example, any category with pullbacks defines (trivially) a display map category, by taking ext to be the identity. The reason why we might want to restrict the class of display maps is that, in some cases, not all morphisms might be exponentiable, which means that the canonical comprehension category structure does not have dependent products (see chapter 2).

Definition 1.6.3. A *contextual category* is a category with attributes, together with a distinguished terminal object \bullet , and a function $\ell : \text{obj } \mathcal{C} \rightarrow \mathbb{N}$ (read “length”), such that:

- (i) \bullet is the unique object of length 0 (on the nose)
- (ii) for any type σ in the context Γ , $\ell(\Gamma.\sigma) = \ell \Gamma + 1$
- (iii) for any non-empty context Γ , there exists a unique context Δ (called the *father* of Γ), and a type σ over Δ such that $\Gamma = \Delta.\sigma$.

The notion of contextual category is the exact semantic counterpart of the syntax of type theory as usually defined (see for example [12]).

The basic idea is that a contextual category is a category with attributes where every context is built uniquely with a finite number of context extensions starting from the empty context.

1.7 Contextualisation

Contextual categories are easier to work with than general (split) comprehension categories, because the existence of the length function allows us to use inductive arguments to prove properties of contexts and morphisms.

Fortunately, most of those arguments work more generally, as long as we carefully restrict them to morphisms that can be built by iterated extensions.

To make this idea precise, we will introduce the notion of *contextualisation* for a general split comprehension category \mathcal{C} .

Definition 1.7.1. A **projection** in \mathcal{C} is a finite sequence of composable display maps.

Contexts and projections form a category which we denote by \mathcal{C}_0 . There is an obvious identity-on-objects functor $i_0 : \mathcal{C}_0 \rightarrow \mathcal{C}$, which maps every projection to the composition of its elements. The length of a projection is defined in the obvious way.

Note that for any context Θ , i_0 can be lifted to the corresponding slice categories, to give a functor $i_0^\Theta : \mathcal{C}_0/\Theta \rightarrow \mathcal{C}/\Theta$.

Definition 1.7.2. Let Θ be a context of \mathcal{C} . The **contextualisation** of \mathcal{C} relative to Θ is defined to be the comma category $(i_0^\Theta \downarrow \text{id}_\Theta)$, and is denoted by $\mathcal{C}^{(\Theta)}$.

Explicitly, objects of $\mathcal{C}^{(\Theta)}$ are the same as the objects of \mathcal{C}_0/Θ , i.e. projections to Θ . Morphisms in $\mathcal{C}^{(\Theta)}$ between two such projections $p : \Gamma \rightarrow \Theta$ and $p' : \Delta \rightarrow \Theta$ are morphisms $f : \Gamma \rightarrow \Delta$ in \mathcal{C} , such that the following diagram

$$\begin{array}{ccc} \Gamma & \xrightarrow{f} & \Delta \\ & \searrow p & \swarrow p' \\ & & \Theta \end{array}$$

is commutative.

There is an obvious fully faithful functor $i^\Theta : \mathcal{C}^{(\Theta)} \rightarrow \mathcal{C}$. Furthermore, we have a function $\ell : \mathcal{C}^{(\Theta)} \rightarrow \mathbb{N}$ which maps every projection to its length.

Proposition 1.7.3. For any context Θ , the contextualisation $\mathcal{C}^{(\Theta)}$ is a contextual category.

Proof. If $p = \text{cod} \circ \text{ext} : \mathcal{E} \rightarrow \mathcal{C}$ is the fibration which is part of the comprehension category structure on \mathcal{C} , we define $\mathcal{E}^{(\Theta)}$ via the pullback:

$$\begin{array}{ccccc} \mathcal{E}^{(\Theta)} & \xrightarrow{j^\Theta} & \mathcal{E} & \xrightarrow{\text{ext}} & \mathcal{C}^I \\ p^\Theta \downarrow & \lrcorner & \downarrow p & & \swarrow \\ \mathcal{C}^{(\Theta)} & \xrightarrow{i^\Theta} & \mathcal{C} & & \end{array}$$

We take id_Θ as the terminal object. There is clearly a unique cartesian functor ext^Θ such that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{E}^{(\Theta)} & \xrightarrow{\text{ext}^\Theta} & (\mathcal{C}^{(\Theta)})^I \\ j^\Theta \downarrow & & \downarrow (i^\Theta)^I \\ \mathcal{E} & \xrightarrow{\text{ext}} & \mathcal{C}^I \end{array}$$

and that defines a split comprehension category structure on $\mathcal{C}^{(\Theta)}$.

The identity projection on Θ is clearly a terminal object in $\mathcal{C}^{(\Theta)}$. All the other axioms of contextual categories are now easy to verify directly. \square

Definition 1.7.4. A morphism $f : \Delta \rightarrow \Gamma$ is said to be **small** if is in the image of i^Θ for some context Θ .

Chapter 2

Dependent sums and products

The simplified definitions of sums and products given in section 1.1 can now be generalised to comprehension categories. In the following, \mathcal{C} will denote a comprehension category, with context extension functor $\text{ext} : \mathcal{E} \rightarrow \mathcal{C}^I$.

Definition 2.0.5. We say that \mathcal{C} has **weak pseudo dependent sums** if, for every type σ over Γ , the substitution functor $p_\sigma^* : \mathcal{E}_\Gamma \rightarrow \mathcal{E}_{\Gamma.\sigma}$ has a left adjoint Σ_σ , and for all morphisms $f : \Delta \rightarrow \Gamma$ in \mathcal{C} , the *Beck-Chevalley condition* is satisfied, i.e. the canonical natural transformation

$$\begin{array}{c}
 \Sigma_{f^*\sigma} q(f, \sigma)^* \\
 \downarrow \Sigma_{f^*\sigma} q(f, \sigma)^* \eta \\
 \Sigma_{f^*\sigma} q(f, \sigma)^* p_\sigma^* \Sigma_\sigma \\
 \downarrow \cong \\
 \Sigma_{f^*\sigma} p_{f^*\sigma}^* f^* \Sigma_\sigma \\
 \downarrow \epsilon_{f^* \Sigma_\sigma} \\
 f^* \Sigma_\sigma
 \end{array} \tag{2.1}$$

is an isomorphism.

Definition 2.0.6. We say that \mathcal{C} has **pseudo dependent sums** if it has weak dependent sums, and for every type σ , the functor Σ_σ is full and faithful.

Definition 2.0.7. We say that \mathcal{C} has **pseudo dependent products** if, for every type σ over Γ , the substitution functor $p_\sigma^* : \mathcal{E}_\Gamma \rightarrow \mathcal{E}_{\Gamma.\sigma}$ has a right adjoint Π_σ , and

for all morphisms $f : \Delta \rightarrow \Gamma$ in \mathcal{C} , the *Beck-Chevalley condition* is satisfied, i.e. the canonical natural transformation

$$\begin{array}{c}
 f^* \Pi_\sigma \\
 \downarrow \eta_{f^* \Pi_\sigma} \\
 \Pi_{f^* \sigma} p_{f^* \sigma} f^* \Pi_\sigma \\
 \downarrow \cong \\
 \Pi_{f^* \sigma} q(f, \sigma)^* p_\sigma^* \Pi_\sigma \\
 \downarrow \Pi_{f^* \sigma} q(f, \sigma)^* \epsilon \\
 \Pi_{f^* \sigma} q(f, \sigma)^*
 \end{array} \tag{2.2}$$

is an isomorphism.

Display map categories admit simple characterisations of pseudo dependent sums and products.

Proposition 2.0.8. *A display map category has pseudo dependent sums if and only if display maps are closed under composition.*

Proposition 2.0.9. *A display map category has pseudo dependent products if and only if all display maps are exponentiable.*

Corollary 2.0.10. *A locally cartesian closed category, regarded as a comprehension category, has pseudo dependent sums and products.*

2.1 Strictification of sums and products

When working with a split comprehension category, the notion of pseudo dependent sum and product is too weak.

Definition 2.1.1. Let \mathcal{C} be a split comprehension category. We say that \mathcal{C} has dependent sums (resp. products) if it has pseudo dependent sums (resp. products) and the isomorphism (2.1) (resp. (2.2)) is the identity.

Note that the definition of dependent sums implies that the following diagram of

functors:

$$\begin{array}{ccc}
 \mathcal{E}_{\Gamma.\sigma} & \xrightarrow{(q f \sigma)^*} & \mathcal{E}_{\Delta.f^*\sigma} \\
 \Sigma_\sigma \downarrow & & \downarrow \Sigma_{f^*\sigma} \\
 \mathcal{E}_\Gamma & \xrightarrow{f^*} & \mathcal{E}_\Delta
 \end{array}$$

is strictly commutative, and similarly for products.

The following result relates pseudo dependent sums in a comprehension category with dependent sums in its strictification.

Proposition 2.1.2. *Let \mathcal{C} be a comprehension category with pseudo dependent sums. Then the strictification of \mathcal{C} has dependent sums.*

Of course, a similar result holds for products, with analogous proof:

Proposition 2.1.3. *Let \mathcal{C} be a comprehension category with pseudo dependent products. Then the strictification of \mathcal{C} has dependent products.*

As an immediate consequence, in the case of locally cartesian closed categories, we get:

Corollary 2.1.4 (Hofmann [6]). *Let \mathcal{C} be a locally cartesian closed category, regarded as a comprehension category. Then the strictification of \mathcal{C} has dependent sums and products.*

Chapter 3

Intensional equality

This chapter will be devoted to studying internal notions of equality in a split comprehension category \mathcal{C} .

The following definition of identity type is limited to split comprehension categories, and is based on the usual syntactic presentation of intensional equality.

Definition 3.0.5. An *identity type structure* on a split comprehension category \mathcal{C} is given by:

- for any type σ in the context Γ , a type Id_σ in the context $\Gamma.\sigma.\sigma$ (we denote by I_σ the context $\Gamma.\sigma.\text{Id}_\sigma$);
- a morphism $r_\sigma : \Gamma.\sigma \rightarrow I_\sigma$, such that the following diagram commutes:

$$\begin{array}{ccc} \Gamma.\sigma & \xrightarrow{r_\sigma} & I_\sigma \\ & \searrow v_\sigma & \swarrow \\ & \Gamma.\sigma.\sigma & \end{array} \quad (3.1)$$

- for any commutative square

$$\begin{array}{ccc} \Gamma.\sigma & \longrightarrow & \Delta.\tau \\ r_\sigma \downarrow & & \downarrow p_\tau \\ I_\sigma & \longrightarrow & \Delta \end{array} \quad (3.2)$$

a diagonal lifting $I_\sigma \rightarrow \Delta.\tau$ that makes both triangles commute.

such that all data is stable under substitutions.

For any terms $\Gamma \vdash M N : \sigma$, we denote by $M \equiv N$ the type over Γ obtained by substituting ld_σ along the morphism $\langle \text{id}, M, N \rangle : \Gamma \rightarrow \Gamma.\sigma.\sigma$. In particular, we have

$$I_\sigma = \llbracket \Gamma(x y : \sigma)(p : x \equiv y) \rrbracket$$

For any term $M : \sigma$ over Γ , we denote $r_\sigma \circ M$ by refl_M . Therefore, refl_M is a term of type $M \equiv M$ over Γ .

It would be useful to give a definition of identity types in the style of those in chapter 2. There is a formulation of identity types as adjoints (namely, left adjoints of the reindexing functor along variables), but this gives *extensional identity types* (see section 3.3). It is not currently clear to the author how to properly weaken the adjunction to get an equivalent formulation to definition 3.0.5.

In the presence of identity types, every type σ can be given a structure of weak ω -groupoid (see [15]).

3.1 Trivial cofibrations

Given an identity type structure on a split comprehension category \mathcal{C} , one can define a weak factorisation system on the syntactic subcategory of \mathcal{C} . This is a slight generalisation of the result in [4].

In the following, \mathcal{C} will denote a split comprehension category with identity types.

Definition 3.1.1. A morphism f is said to be a **trivial cofibration** if it has the left lifting property with respect to all display maps.

The choice of terminology is motivated by the fact that, assuming the existence of some extra structure on \mathcal{C} , one could define a pre-model category structure where the trivial cofibrations correspond to the class of morphisms that we just defined. Here we will only deal with the “right” weak factorisation system, and refer to [11] for more details.

Proposition 3.1.2. *For any type σ , $r_\sigma : \Gamma \rightarrow I_\sigma$ is a trivial cofibration.*

Proof. Immediate from the definition of identity type structure. □

Proposition 3.1.3 (Paulin-Mohring rule). *Let Γ be a context, σ a type over Γ , and M a term of type σ . Denote by $I_{\sigma, M}$ the based path space:*

$$\llbracket \Gamma(x : \sigma)(p : x \equiv M) \rrbracket$$

The morphism $\Gamma \rightarrow I_{\sigma, M}$ given by

$$\begin{aligned} x &\mapsto M \\ p &\mapsto \text{refl}_M \end{aligned}$$

is a trivial cofibration.

Lemma 3.1.4. *Every small morphism f can be factored as $p \circ i$, where p is a composition of weakening morphisms, and i is a trivial cofibration.*

Proof. Since f is small, we can assume that \mathcal{C} is a contextual category.

We proceed by induction on the length of the target of f . If $f : \Gamma \rightarrow \bullet$, then f is a projection, so we take the identity as i and f itself as p .

Now, assuming that we have a factorisation $f = p \circ i : \Gamma \rightarrow \Delta$, we will exhibit a factorisation for $q(f, \tau)$, where τ is a type over Δ . Since $q(f, \tau) = q(p, \tau) \circ i$, it is enough to suppose that f is itself a composition of weakening morphisms.

We have that $q(f, \tau) = \langle f, M \rangle$, for some term $M : f^*\tau$ over Γ , so we can find a factorisation

$$\Gamma \xrightarrow{r_{f^*\tau, M}} I_{f^*\tau, M} \xrightarrow{q} \Delta.\tau$$

where q is obtained from

$$I_{f^*\tau, M} \longrightarrow \Gamma \xrightarrow{f} \Delta$$

by extending along the variable x of $I_{f^*\tau, M}$.

To conclude, observe that q is obtained by extension from a composition of weakenings, while $r_{\sigma, M}$ is a trivial cofibration by proposition 3.1.3. \square

3.2 Weak equivalences

Definition 3.2.1. Let $f, g : \Gamma \rightarrow \Delta$ be two morphisms, and $p : \Delta \rightarrow \Theta$ a projection. We define a p -homotopy between f and g by induction on the length of p :

- if $\Delta = \Theta$, a homotopy between f and g is just an equality $f = g$
- if $\Delta = \Delta_0.\sigma$, $f = \langle f_0, M \rangle$ and $g = \langle g_0, N \rangle$, and h is a homotopy between f_0 and g_0 , we use h to get a morphism $s_h : f_0^*\sigma \rightarrow g_0^*\sigma$, and define a homotopy between f and g as a pair (h, p) , where $p : M \equiv s_h^*N$.

Definition 3.2.2. Two morphisms are said to be **homotopic** if there exists a p -homotopy between them for some projection p .

Definition 3.2.3. A morphism $f : \Delta \rightarrow \Gamma$ is a **weak equivalence** if there exists $g : \Gamma \rightarrow \Delta$ such that $g \circ f$ and $f \circ g$ are both homotopic to identities.

Proposition 3.2.4. A small trivial cofibration is a weak equivalence.

3.3 Extensional equality

Definition 3.3.1. An identity type structure on \mathcal{C} is said to be *extensional* if for any context Γ and any type σ in Γ ,

$$r_\sigma : \Gamma.\sigma \rightarrow I_\sigma$$

is an isomorphism.

Proposition 3.3.2. Let \mathcal{C} be a split comprehension category with an identity type structure. The following are equivalent.

- (i) the identity type structure is extensional
- (ii) the lifting problem 3.2 for any display map has a unique solution
- (iii) in every context Γ , $\Gamma \vdash P : M \equiv N$ implies $\Gamma \vdash M = N$
- (iv) all weak equivalences are isomorphisms
- (v) every small morphism is a fibration

Proof. **(i) \Rightarrow (ii)** Obvious.

(ii) \Rightarrow (iii) The two variables x and y in I_σ are both valid diagonal liftings in the following diagram:

$$\begin{array}{ccc} \Gamma.\sigma & \longrightarrow & \Gamma.\sigma \\ \downarrow & & \downarrow \\ I_\sigma & \longrightarrow & \Gamma \end{array}$$

thus they are equal. Composing with the morphism $\Gamma \rightarrow I_\sigma$ given by

$$\begin{cases} x \mapsto M \\ y \mapsto N \\ p \mapsto P \end{cases}$$

we then get that $M = N$.

(iii)⇒(iv) If the existence of a term of type $M \equiv N$ implies that $M = N$, the existence of a homotopy between two morphisms f and g implies that $f = g$. Thus weak equivalences are the same as isomorphisms.

(iv)⇒(v) Let f be a small morphism. Factor f as a small trivial cofibration i followed by a fibration p . Since i is a weak equivalence, it is an isomorphism, so f is isomorphic to p , hence a fibration itself.

(v)⇒(i) The morphism r_σ is clearly small, thus it is a fibration by assumption. Since it is a trivial cofibration by proposition 3.1.2, it follows that it is an isomorphism.

□

Lemma 3.3.3. *Suppose \mathcal{C} has extensional identity types. Then for every term $\Gamma \vdash M : \sigma$, refl_M is the only term of type $M \equiv M$.*

Proof. Let us work in the context $I_\sigma = \llbracket \Gamma(x, y : \sigma)(p : x \equiv y) \rrbracket$. By 3.3.2, $x = y$, hence the type of p is $x \equiv x$, so refl_x has the same type as p . We can then consider the type $p \equiv \text{refl}_x$ in the context I_σ .

We have the following commutative diagram:

$$\begin{array}{ccc} \Gamma.\sigma & \longrightarrow & \llbracket I_\sigma.(q : p \equiv \text{refl}_x) \rrbracket \\ r_\sigma \downarrow & & \downarrow \\ I_\sigma & \xrightarrow{\text{id}} & I_\sigma \end{array}$$

where the top arrow is given by: $(x \mapsto x; y \mapsto x; p \mapsto \text{refl}_x; q \mapsto \text{refl}_{\text{refl}_x})$.

The diagonal lifting gives a term $I_\sigma \vdash Q : p \equiv \text{refl}_x$, so by 3.3.2 again, we conclude that $p = \text{refl}_x$.

Now, given any term $\Gamma \vdash P : M \equiv M$, substituting along the morphism $\Gamma \rightarrow I_\sigma$ given by $(x \mapsto M; y \mapsto M; p \mapsto P)$, we get that $P = \text{refl}_M$, so refl_M is the only term of its type. □

For extensional identity types, we can prove a result like Corollary 2.1.4:

Proposition 3.3.4 (Hofmann [6]). *A locally cartesian closed category regarded as a split comprehension category has extensional identity types.*

Proof (sketch). Types over Γ are cartesian functors $\text{dom}_\Gamma \rightarrow p$, where p is the codomain fibration.

Given such a type σ , we define Id_σ as follows. Consider any $g : \Delta \rightarrow \Gamma.\sigma.\sigma$, and write $g = \langle f, M, N \rangle$ for some $f : \Delta \rightarrow \Gamma$, and terms M, N of type σf over Δ .

The value of Id_σ on f is then defined to be the equalizer of M and N . □

Chapter 4

Universes

In the following, we will be working in a split comprehension category \mathcal{C} with a terminal object \bullet .

Definition 4.0.5. A **universe** in \mathcal{C} is given by a type \mathcal{U} in the empty context, and a type El in the context $\bullet.\mathcal{U}$.

We use \mathcal{U} to denote the context $\bullet.\mathcal{U}$ as well, and $\tilde{\mathcal{U}}$ for $\mathcal{U}.\text{El}$.

If \mathcal{C} is a locally cartesian closed category, equipped with the strict comprehension category structure given by proposition 1.5.2, a universe in \mathcal{C} is a morphism

$$\tilde{\mathcal{U}} \rightarrow \mathcal{U},$$

together with a choice of pullbacks for any morphism $X \rightarrow \mathcal{U}$. This coincides with the definition of universe given in [10].

The reason for considering universes is that they allow us to build a full model of intensional type theory (as a split comprehension category) within them, and most of the construction can be done in the internal type theory.

Given a universe, and a morphism $f : \Gamma \rightarrow \mathcal{U}$, we denote by σ_f the substitution of El along f .

Now we can define a category with attributes structure on \mathcal{C} (which we denote $\mathcal{C}_{\mathcal{U}}$, and call the *restricted model* determined by \mathcal{U}) with a representable Ty functor as follows:

$$\begin{aligned} \text{Ty } \Gamma &= \mathcal{C}(\Gamma, \mathcal{U}) \\ \text{ext } f &= \text{ext } \sigma_f \end{aligned}$$

4.1 Type formers in a restricted model

If a split comprehension category \mathcal{C} has dependent products, there is a very simple *internal* criterion to determine whether a restricted model determined by a given universe \mathcal{U} inherits the dependent product structure.

Similar considerations apply to other type formers, but we will focus on dependent types in the following. Since we are working with split comprehension categories, we can use type theoretic notation, as detailed in section 1.4.

To distinguish between the original model \mathcal{C} and the restricted model \mathcal{C}_U , we will write judgements in \mathcal{C} in **blue**, and judgements in \mathcal{C}_U in **green**.

To avoid confusion between the meta-theoretic Π term introduced below, and the internal notation for dependent products, we will use Agda-style syntax: a type of the form $\Pi_{\sigma}\tau$ will be denoted as $(x : \sigma) \rightarrow \tau(x)$. Finally, the empty context \bullet will usually be omitted from the notation.

Definition 4.1.1. We say that a universe \mathcal{U} has dependent products if there is a term

$$\Pi : (\sigma : \mathcal{U}) \rightarrow (El \sigma \rightarrow \mathcal{U}) \rightarrow \mathcal{U}$$

such that, in the context $\llbracket (\sigma : \mathcal{U})(\tau : El(\sigma) \rightarrow \mathcal{U}) \rrbracket$, we have:

$$El (\Pi \sigma \tau) = (x : El \sigma) \rightarrow El(\tau x) \tag{4.1}$$

Now we can show how dependent products for \mathcal{U} induce dependent products for the restricted model \mathcal{C}_U . For clarity, we will follow the conventional rule-based approach (see [7] for details). It is not hard to show that this really gives dependent products in the sense of definition 2.1.1.

Formation

Given judgements in \mathcal{C}_U :

$$\begin{aligned} &\Gamma \vdash \sigma \text{ type} \\ &\Gamma(x : \sigma) \vdash \tau(x) \text{ type} \end{aligned}$$

These correspond to judgements in \mathcal{C} :

$$\begin{aligned} &\Gamma \vdash \sigma : \mathcal{U} \\ &\Gamma(x : El\sigma) \vdash \tau(x) : \mathcal{U} \end{aligned}$$

From the second of which we get

$$\Gamma \vdash \tau : (x : \text{El } \sigma) \rightarrow \mathcal{U}$$

Now we can apply Π :

$$\Gamma \vdash \Pi \sigma \tau : \mathcal{U}$$

And we obtain, by substituting into equation (4.1):

$$\text{El } (\Pi \sigma \tau) = (x : \text{El } \sigma) \rightarrow \text{El } (\tau x)$$

So, we take $\Pi \sigma \tau$ as the product of σ and τ .

Introduction

Let σ and τ as before, and fix a term

$$\Gamma(x : \sigma) \vdash M : \tau x$$

in $\mathcal{C}_{\mathcal{U}}$. This corresponds to a term in \mathcal{C} :

$$\Gamma(x : \text{El } \sigma) \vdash M : \text{El}(\tau x)$$

abstracting, we get

$$\Gamma \vdash \lambda M : (x : \text{El } \sigma) \rightarrow \text{El}(\tau x)$$

which, reinterpreted in $\mathcal{C}_{\mathcal{U}}$ gives:

$$\Gamma \vdash \lambda M : \Pi \sigma \tau$$

So, abstraction for $\mathcal{C}_{\mathcal{U}}$ strictly coincides with abstraction in \mathcal{C} . Stability under substitution follows easily.

Elimination

A similar argument works for elimination. Suppose we have (in $\mathcal{C}_{\mathcal{U}}$):

$$\Gamma \vdash M : \Pi \sigma \tau$$

$$\Gamma \vdash N : \sigma$$

We get, in \mathcal{C} :

$$\begin{aligned}\Gamma \vdash M &: (x : \text{El } \sigma) \rightarrow \text{El } (\tau x) \\ \Gamma \vdash N &: \text{El } \sigma\end{aligned}$$

therefore, by application:

$$\Gamma \vdash M N : \text{El } (\tau N)$$

We know, by β applied to τ , that

$$\tau N = \tau(N)$$

so

$$\Gamma \vdash M N : \text{El } (\tau(N))$$

which gives, back in $\mathcal{C}_{\mathcal{U}}$:

$$\Gamma \vdash M N : \tau(N)$$

Again, the fact that elimination is stable under substitution follows from the corresponding fact for \mathcal{C} .

The β and η rules follow immediately from the corresponding rules for \mathcal{C} .

Chapter 5

Examples

5.1 Set model

The category of sets is locally cartesian closed by proposition 1.2.3. Therefore, the corresponding split comprehension category has dependent sums and products (Corollary 2.1.4) and extensional identity types (proposition 3.3.4).

Assuming a set-theoretic foundation, every regular cardinal κ determines a universe \mathcal{U}_κ in \mathbf{Set} , regarded as a split comprehension category.

Namely, the cumulative hierarchy at level κ (see for example [9]) is a locally cartesian closed subcategory V_κ of \mathbf{Set} , hence dependent sums, products and identity types are inherited by the restricted model $\mathbf{Set}_{\mathcal{U}_\kappa}$.

5.2 Groupoid model

The first example of a model of intensional (i.e. non-extensional) identity types was given in [8] using the category \mathbf{Gpd} of groupoids.

The basic idea is to take fibrations of groupoids (that is, Grothendieck fibrations of the underlying categories) as types.

Using theorem 1.3.17 one can show that a morphism of groupoids is exponentiable if and only if it is a fibration, thus fibrations determine a display map category with dependent products by proposition 2.0.9.

Dependent sums follow easily from proposition 2.0.8 and the fact that fibrations are closed under composition.

The most interesting part of the construction is, of course, the definition of identity types. For simplicity, we only carry it out for the empty context, referring to the original paper [8] for full details.

Since every morphism to the one-point groupoid is automatically a fibration, types over the empty context can be identified with groupoids themselves. Given a groupoid A , we define the identity type for it to be the obvious map $p_{\text{id}_A} : A^I \rightarrow A \times A$, where A^I is the *arrow category* of A .

Of course, A^I is itself a groupoid, and it is not hard to see that p_{id_A} is a fibration. The morphism $r_A : A \rightarrow A^I$ is defined by $r_A(a) = \text{id}_a$, and it clearly makes the diagram (3.1) commute.

To show that the lifting problem (3.2) has always a solution, it is enough to find a diagonal lifting g for the following diagram:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow & & \downarrow p \\ A^I & \xrightarrow{=} & A^I, \end{array}$$

where p is any fibration over A^I .

Given an object $\alpha : x' \rightarrow x$ in A^I , we consider the following commutative diagram in A

$$\begin{array}{ccc} x & \xrightarrow{\text{id}} & x \\ \text{id} \downarrow & & \downarrow \alpha \\ x & \xrightarrow{\alpha} & y \end{array}$$

regarded as a morphism $u : \text{id} \rightarrow \alpha$ in A^I , and apply the lifting property of the fibration p to u , to get an object $g\alpha$ of B above α .

This gives a definition of g for objects of A^I . It is not hard to extend g to morphisms and prove all the required properties.

5.3 Presheaf models

Let \mathcal{D} be any small category, and define \mathcal{C} as the category of Set-valued functors (*presheaves*) on \mathcal{D} .

Proposition 5.3.1. *The category \mathcal{C} is locally cartesian closed.*

Proof. It is well known that any presheaf category is cartesian closed. To show that every slice is also cartesian closed, it suffices to observe that, for any presheaf X :

$$\text{Set}^{\mathcal{D}}/X \cong \text{Set}^{f X}$$

□

Therefore, as in the case of Set , we get a comprehension category structure on \mathcal{C} with dependent products, sums, and extensional identity types.

In the following, we define a universe for \mathcal{C} . The basic idea is taken from [10], but the details of the following construction are original, and do not require the axiom of choice.

Let \mathcal{C}' be category of functors $\mathcal{D} \rightarrow \text{Cat}$, together with *lax* natural transformations. The functor $\delta : \text{Set} \rightarrow \text{Cat}$, which gives the discrete category over a set, induces a functor $\delta_* : \mathcal{C} \rightarrow \mathcal{C}'$ by precomposition.

Lemma 5.3.2. *The functor δ_* has a right adjoint ε .*

Proof. The category \mathcal{C} is locally finitely presentable, and \mathcal{C}' is locally small, hence it is enough to show that δ_* preserves small colimits. So, let $(X^i)_{i:I}$ be a small diagram of presheaves, and $\nu^i : X^i \rightarrow L$ its colimit cocone. We want to show that it is a universal cocone in \mathcal{C}' as well.

To that end, suppose that $\eta^i : X^i \rightarrow Z$ is another cocone, where each η^i is a lax natural transformation. For every $n : \mathcal{D}$, since colimits commute with functor application, we get a function $\eta_n : L_n \rightarrow Z_n$, such that for all $i : I, x : X_n^i$

$$\eta_n x = \eta_n^i(\nu_n^i x). \quad (5.1)$$

All we have to do now is define η on morphisms of \mathcal{D} . So let $d : n \rightarrow m$ be such a morphism, and let $x : X_m$. Choose an $i : I$ and $\tilde{x} : X_m^i$ such that $x = \nu^i \tilde{x}$, and define $\eta_d x = \eta_d^i \tilde{x}$, which has the correct type thanks to 5.1, and is well defined because the η_i form a cocone.

It is easy to see that η is a lax natural transformation, and that it is unique. Thus L is the colimit of the X^i in \mathcal{C}' . □

Now let V be a small cartesian closed category of sets (for example, V_κ for some regular cardinal κ). We denote by W the image under ε of the constant functor on

V . The adjunction between δ_* and ε , gives, for any presheaf X , a natural isomorphism:

$$\mathcal{C}(X, W) \cong \mathcal{C}'(X, \text{Const } V). \quad (5.2)$$

The idea of this construction is that lax natural transformations between a presheaf X and the constant functor V classify presheaves over X with fibers in V , so W is the classifying presheaf for such maps, which we will call V -small.

In fact, let $\eta : \Gamma \rightarrow V$ be lax natural transformation. For $n : \mathcal{D}$, define

$$E_n^\eta = \sum (x : \Gamma_n) . \eta_n x,$$

and for $d : n \rightarrow m$,

$$\begin{aligned} d^* : E_m^\eta &\rightarrow E_n^\eta \\ d^*(x, y) &= (d^*x, \eta_d x y). \end{aligned}$$

Clearly, the first projection $E^\eta \rightarrow \Gamma$ is a natural transformation, so E^η can be regarded as an element of \mathcal{C}/Γ .

Vice versa, every V -small $E \rightarrow \Gamma$ is naturally isomorphic over Γ to E^η for some lax natural transformation $\eta : \Gamma \rightarrow V$.

Furthermore, for any $f : \Delta \rightarrow \Gamma$, we have an obvious pullback diagram:

$$\begin{array}{ccc} E^{\eta \circ f} & \longrightarrow & E^\eta \\ \downarrow & & \downarrow \\ \Delta & \longrightarrow & \Gamma \end{array}$$

In 2-categorical language, E defines a pseudonatural transformation between $\mathcal{C}'(-, V)$ and the pseudofunctor $S : \Gamma \mapsto \mathcal{C}/\Gamma$ whose essential image is the pseudo-subfunctor of S consisting of V -small maps.

Now, let $w : W \rightarrow V$ be the counit of the adjunction 5.2. Then E^w is a V -small map:

$$p_w : \widetilde{W} \rightarrow W,$$

which, together with the choice of pullbacks given by E above, defines a universe for \mathcal{C} .

5.4 Models in strict ω -categories

This section is a sketch of a model construction based on strict ω -categories, currently a work in progress. The following uses the terminology of Homotopy Type Theory, for which we refer the reader to [13].

In [10], the authors construct a contextual category using Kan fibrations of simplicial sets, with dependent sums, products, intensional identity types, and a universe satisfying the univalence axiom.

Section 5.3 above can be regarded as a first step in the construction of the univalent universe.

This was not the first example of such a model. In fact, the groupoid model in [8] already has this property. What really sets the simplicial model apart from previous constructions is that the univalent universe is not limited to n -truncated types for some n , but contains types of arbitrarily large h-level, and even types with no h-level at all.

Unfortunately, models based on simplicial sets have so far proved very hard to internalise in type theory, and thus it is still not possible to extract a decidable implementation from them.

A recent attempt [1] based on *semi*-simplicial sets seems promising, but a lot of problems remain.

Given the evident combinatorial difficulties of weak ω -groupoids, one could ask whether using *strict* ω -groupoids, it would be easier to find a solution. Indeed, a model of type theory based on strict ω -groupoids has already been developed [16].

Unfortunately, it is quite easy to convince oneself that such a model has no chance of admitting a univalent universe consisting of more than sets (i.e. 0-truncated types).

Intuitively, in fact, in a universe containing, say, 1-truncated types (groupoids), the identity types would correspond to isomorphisms (since the universe itself must be a strict ω -groupoid), but weak equivalences correspond to equivalences of groupoids.

This is the same problem that the original groupoid model has. Introducing ω -groupoids allows one to consider higher types, but univalence still only holds up to sets.

Therefore, our approach is to work with strict ω -categories with (chosen) weak

inverses. In an effort keep the combinatorial complexity tame, we also require the weak inverse operator to be involutive. We call these structures *pseudogroupoids*.

To see why it is reasonable to expect this model to admit a univalent universe, we take the set of all “small” pseudogroupoids (e.g. in some V_κ), and show how we can equip it with a structure of pseudogroupoid in such a way that weak equivalences correspond to morphisms.

Since pseudogroupoids are in particular weak ω -groupoids, we can regard them as Kan complexes, and therefore work internally within the simplicial model.

Given small pseudogroupoids A and B , we define a morphism in the universe between A and B to be an infinite tower of functions in both directions like the following:

$$\begin{aligned}
f_0 &: A \rightarrow B \\
g_0 &: B \rightarrow A \\
f_1 &: (x : A)(y : B) \rightarrow f_0(x) \equiv y \rightarrow x \equiv g_0(y) \\
g_1 &: (x : A)(y : B) \rightarrow x \equiv g_0(y) \rightarrow f_0(x) \equiv y \\
f_2 &: \forall x y (x_1 : f_0(x) \equiv y)(y_1 : x \equiv g_0(y)) \\
&\quad \rightarrow f_1(x_1) \equiv y_1 \rightarrow x_1 \equiv g_1(y_1) \\
g_2 &: \forall x y (x_1 : f_0(x) \equiv y)(y_1 : x \equiv g_0(y)) \\
&\quad \rightarrow x_1 \equiv g_1(y_1) \rightarrow f_1(x_1) \equiv y_1 \\
&\dots
\end{aligned}$$

It is not hard to see that these towers give a strict ω -category structure on the universe. To show that it forms a pseudogroupoid, one has to define an involutive symmetry, which seems reasonably easy to do, since this formulation of equivalence is manifestly symmetric (as opposed to, say, the original one by Voevodsky saying that all fibres are contractible).

A lot of work is needed to turn this idea into a fully rigorous construction. The involutive inverses make many of the details easier to work out, but they are still quite challenging.

Furthermore, the infinite representation for equivalences would present additional challenges, if one were to internalise the model in a pre-existing type theory.

Bibliography

- [1] B. Barras, T. Coquand, and S. Huber. A generalization of takeuti-gandy interpretation. 2013.
- [2] F. Borceux. *Handbook of Categorical Algebra*, volume 2. 1994.
- [3] F. Conduché. Au sujet de l'existence d'adjoints à droite aux foncteurs "image réciproque" dans la catégorie des catégories. 1972.
- [4] N. Gambino and R. Garner. The identity type weak factorisation system. 2008.
- [5] J. Giraud. *Méthode de la descente*. 1964.
- [6] M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *Proceedings of Computer Science Logic, Lecture Notes in Computer Science*, pages 427–441. Springer, 1994.
- [7] M. Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.
- [8] M. Hofmann and T. Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.
- [9] T. Jech. *Set Theory*. 2003.
- [10] C. Kapulkin, P. L. Lumsdaine, and V. Voevodsky. The simplicial model of univalent foundations. 2012.
- [11] P. L. Lumsdaine. Model structures from higher inductive types. 2011.
- [12] P. Martin-Löf. *Intuitionistic type theory*. 1980.
- [13] The Univalent Foundation Program. *Homotopy Type Theory*.

- [14] R. A. G Seely. Locally cartesian closed categories and type theory. 1983.
- [15] B. van den Berg and R. Garner. Types are weak omega-groupoids. 2008.
- [16] M. A. Warren. The strict ω -groupoid interpretation of type theory. 2000.