University of Nottingham

School of Computer Science

Models of Type Theory with Strict Equality

Paolo Capriotti

A thesis submitted to the University of Nottingham for the degree of

Doctor of Philosophy

July 2016

Abstract

This thesis introduces the idea of *two-level type theory*, an extension of Martin Löf type theory [27] that adds a notion of *strict equality* as an internal primitive.

A type theory with a strict equality alongside the more conventional form of equality, the latter being of fundamental importance for the recent innovation of *homotopy type theory* (HoTT), was first proposed by Voevodsky [38], and is usually referred to as HTS.

Here, we generalise and expand this idea, by developing a semantic framework that gives a systematic account of type formers for two-level systems, and proving a conservativity result relating back to a conventional type theory like HoTT.

Finally, we show how a two-level theory can be used to provide partial solutions to open problems in HoTT. In particular, we use it to construct semi-simplicial types, and lay out the foundations of an internal theory of $(\infty, 1)$-categories.

Acknowledgements

I would like to thank the members of the Functional Programming Laboratory of the University of Nottingham, and in particular my supervisor Venanzio Capretta, for their advice and support, and for providing an excellent and stimulating research environment.

Thorsten Altenkirch, Nicolai Kraus and Christian Sattler deserve particular thanks for countless discussions, type theory meetings, and reading groups, all of which made the time of my PhD extremely enjoyable and fruitful.

And most of all, I would like to express my gratitude to my wife, Elisa, without whose support and encouragement, I would have certainly not made it to this point. Thank you!

CONTENTS

1 INTRODUCTION

Type theory is a foundational framework for mathematics which can also be regarded as a programming language. The central concept of type theory is of course that of a *type*: an entity that plays the double role of a logical statement (a "proposition") and of a collection (a "set").

The sort of type theory developed in this thesis is more specifically referred to as *Martin-Löf dependent type theory* [27], because it is based on the idea that types can depend on values. This, together with a few basic primitives, makes the corresponding calculus powerful enough to express most fundamental mathematical ideas, including universal and existential quantification, functions, ordered pairs, etc.

In recent years, a new branch of type theory, called *homotopy type theory* (HoTT) (section 1.5) has arisen. The main revolution of HoTT consists in embracing the higher dimensional structure of equality, and using it to interpret types not just as sets, but as *topological spaces* up to homotopy equivalence.

This has made it possible to formalise classical results of homotopy theory *synthetically*, that is, without reference to the underlying representation of topological spaces - be it as sets equipped with a collection of open subsets, or any other formulation, possibly more well-behaved in a constructive setting. Instead, spaces are studied abstractly, their features and properties derived simply from those of the types that represent them. This has made the formulations and proofs of homotopical facts extremely elegant and streamlined, cast new light on seemingly well-understood results, and suggested new directions of research.

The interest of homotopy type theory lies in the fact that, by using its underlying type theoretic language, one is restricted to constructions that are *automatically* homotopy-invariant: any concept, or definition, or result, by the mere fact of having been expressed "internally", is guaranteed to remain valid when spaces are replaced with equivalent ones.

This fundamental homotopy-invariance property crystallises into the principle of *univalence*, probably the most important technical innovation of HoTT, which roughly states that (homotopy) equivalent types are *equal*. Equality here is not meant in a *strict* sense (i.e. equal types will not be interpreted as the same object in a model), but rather as the existence of some kind of "path" in the universe connecting the two types.

If follows that equality, despite still adhering to its defining property of being preserved by *all* constructions, is not a *mere proposition* anymore: it may possess non-trivial structure. Paths themselves form a type, and their notion of equality is also subject to the same considerations. Here we see how directly some of the most familiar constructions in classical homotopy theory, such as *homotopy groups*, arise internally in the language of HoTT.

Unfortunately, HoTT, and its homotopy invariant nature, impose some fundamental constraints on the kind of constructions that we are allowed to perform internally. Any classical definition whose (possibly ultimately irrelevant) details depend on more than the homotopy type of the spaces involved, needs to be reworked to fit into the framework of HoTT.

Sometimes, of course, this is not possible, as not all classical results hold in HoTT (a famous example is Whitehead's theorem, that only holds for truncated types in HoTT [36]). Other times, it looks as though it *should* be possible to provide an internal analogue of a classical notion, but all naive attempts fail.

The most prominent example of such a notion is that of *semi-simplicial types*, which we explain in section 1.6. Giving a satisfactory account of this and similar "infinite coherence" problems is the main motivation behind this thesis.

1.1 OVERVIEW

Type theory, especially if directly introduced as a mathematical foundation, is usually presented as a collection of *inference rules*. Where the usual foundations of mathematics are based on some (often not clearly specified) form of first order logic, on top of which the well-known axioms of Zermelo-Fraenkel set theory [41] are laid out, the rules of type theory form a single corpus that describes both the logical and the set-theoretical aspects of mathematics (and much more, as will be clear later when we will describe homotopy type theory).

In this thesis, we follow a slightly unconventional path: we define an algebraic notion of *model of type theory*, as a category equipped with the logical structure necessary to talk about types. The *syntax* of type theory, then, instead of being implicitly defined by a set of rules, is taken to be the *initial* model in our setting.

The advantage of our approach is that we do not have to deal with all the syntactical complications of name binding, type derivations or congruence rules for definitional equality (see for example [18]). In fact, the initial model (provided it exists), is in particular *a* model, hence it comes equipped with all the structure and satisfies all the axioms that we require. Furthermore, and even more obviously, there is no "initiality theorem" [34] to be proved, as the syntax is initial *by definition*.

The disadvantage is that, since the syntax doesn't natively possess a notion of name binding, writing out terms explicitly in the language of the model can be cumbersome, and it makes for expressions that are extremely hard to read. We will subvert this issue by devising a number of notational conventions (sections 2.1.1 and 2.1.7) that will make it possible to work in models of type theory *as if* they had name binding, making constructions in a generic model essentially indistinguishable from their completely syntactical counterparts.

The thesis consists of an introductory chapter (chapter 1), followed by three main chapters. In chapter 2 we lay out our algebraic approach to models of type theory. In chapter 3, we extend our framework to cover *two-level* models and prove a conservativity result. Finally, in chapter 4, we fix a particular instance of two-level type theory, and give some examples of what can be achieved by working internally in such a theory.

Our definition of model of type theory is based on *categories with families* (CwF, [11]), although our definition differs slightly from the original in non-essential ways (section 2.1). We introduce and motivate a number of *basic type formers* (section 2.1.3) using presheaves, and the fact that presheaf categories have a natural CwF structure (section 2.1.2).

Once we have enough basic type formers under our belt, we will give a general definition of "type former" (section 2.2), and show how the basic ones defined previously can also be regarded as instances of the general definition.

We will then introduce two-level models of type theory, where two different type theories are combined in a single system. This kind of structure naturally arises when studying certain homotopical models of type theory: types can be divided into *fibrant* and *strict*, resulting in two "parallel" type theories, with possibly different sets of type formers.

Perhaps surprisingly, with enough assumptions on the type formers involved, a two-level type theory is *conservative* over its fibrant fragment (section 3.2.3), meaning that proofs and constructions using the full two-level theory can always be reworked so that they only use the fibrant fragment, as long as the end result is itself fibrant. The proof mimics that of a similar result on the conservativity of the Logical Framework [18].

The idea of the conservativity proof is straightforward, but is unfortunately made complicated by issues of strictness of coercion of fibrant types into strict types (section 3.2). We work around the strictness issues by defining the notion of *regularity* for models (section 3.2.2).

Finally, we will move completely inside a two-level model, and work in the internal language of the corresponding type theory (chapter 4), in the style of

[36]. We choose a two-level type theory inspired by Voevodsky's HTS [38], but more minimalistic (section 4.1.1).

In our flavour of two-level type theory, we develop the notion of Reedy fibrant diagram, and show how they can be classified by fibrant types. In particular, this yields a definition of *semi-simplicial type*, a notion that has so far eluded all attempts at formalisation in conventional HoTT.

Our construction resembles the one in [17], however, in the latter, a specific consequence of the existence of strict equality has to be assumed in order for the construction to go through. We, instead, build on the general idea of Reedy fibrancy, and make no ad-hoc assumption beyond the general setup of two-level type theory.

From that, we lay out the foundations of an internal development of higher category theory, starting from the definition of *complete semi-Segal type* (definition 4.6.8), and showing why this is a good candidate for a notion of category that is powerful enough to include all the reasonable "categorical" structures present in HoTT, while at the same time allowing all the familiar categorical constructions to be performed within the constraints of type theory.

Most of the mathematical content of this thesis is based on a constructive metatheory. We do not make use of classical principles like the law of excluded middle or the axiom of choice. One exception is the overview of the simplicial model of HoTT given in section 3.1, since the construction referenced in [21] is explicitly non-constructive. [1]

1.2 CONTRIBUTIONS

The main contributions of this thesis are as follows:

- We develop a systematic and generic theory of *type formers*: a single notion that can be instantiated to cover all known examples of what are usually referred to as type formers. This is inspired by the ideas of the *Logical Framework* [16], but our presentation is completely semantic in nature, and can be used to state and prove metatheoretical results about models of type theory without fixing a particular set of type formers in advance.
- We define the notion of *two-level type theory*, making precise and generalising the ideas underlying the HTS theory proposed by Voevodsky [38]. We prove a conservativity result, which implies, among other things, that two-level type theory can be used as a "schematic" language for working with infinite families of objects in a conventional type theory.

1 There do exist attempts at building models of HoTT in a constructive setting [14] [7], but they are still relatively incomplete and poorly understood, hence we do not rely on them in this thesis.

- We show how a particular minimalistic flavour of two-level type theory, similar to HTS, can be used to give partial solutions to some of the most pressing open problems in HoTT. In particular, we give a definition of semi-simplicial type, and use it to lay out the foundation of an internal theory of $(\infty, 1)$-categories in type theory.

In particular, this thesis contains proofs of the following results:

- theorem 3.2.4, showing that any type former on a CwF can be lifted to the *fibrant universe* of its presheaf category;
- theorems 3.2.9 and 3.2.13, drawing a correspondence between a *regular* model of type theory and the two-level model given by its presheaf category;
- theorem 3.2.14, providing a way to prove statements in HoTT using a two-level system;
- theorem 4.5.4, showing how to construct a Reedy fibrant replacement for any inverse diagram in a two-level system;
- theorem 4.3.1, exhibiting an inconsistency of a general *fibrant replacement* operator in a two-level system with non-0-truncated fibrant types (see section 1.5).

1.2.1 *Declaration of authorship*

Section 1.4 and section 2.1 contain background material about semantic models of type theory. Most of the definitions and results of these sections can be found in the literature, but their presentation has been reworked to fit with the constructions introduced later.

Most of the material of chapter 4 is joint work with Thorsten Altenkirch and Nicolai Kraus. The definition of semi-simplicial types and, more generally, Reedy fibrant diagrams, and most of the preliminary content leading up to that, including parts of section 1.6, have been published in [4].

The rest of the thesis is original work of the present author.

1.3 RELATED WORK

The main ideas of this thesis are inspired by Voevodsky's proposal of a *homotopy type system* (HTS), which can be found in [38].

In [8], the authors present a version of a two-level type theory with a fibrant replacement operator, which would be inconsistent in the formulation of this thesis (theorem 4.3.1), to derive a model structure on the universe of strict types.

A two-level type theory is developed in [26]. Their motivation, however, is substantially different, hence the resulting theory has little resemblance with the two-level type theory developed in this thesis.

A lot of work from several authors has recently gone into trying to develop a systematic and rigorous framework for working with models of type theory. Chapter 2 contains one such (partial) attempt. Similar work going in the same general direction can be found in [1, 3, 29, 39, 40].

1.4 FUNDAMENTS OF TYPE THEORY

To motivate the definitions of chapter 2 we will begin by exploring the basic concepts of intuitive type theory, and show how their desired properties translate directly into categorical structures.

1.4.1 *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.

1.4.2 *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,

$$\begin{cases} n \mapsto m \\ R \mapsto k \\ M \mapsto k^m \end{cases} \tag{1}$$

This would define a morphism from Δ to Γ in the category $\mathcal{C}$. It will be clear in chapter 2, once we have a complete definition of CwF, how to make morphism definitions like (1) precise.

The category $\mathcal{C}$ should have a (distinguished) terminal object 1. We call 1 the *unit 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 unit context in any other context.

1.4.3 *Types*

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

- the *context extension* $\Gamma.A$, which is to be thought of as the result of adding a new variable of type A to the existing context Γ
- the *display map* $p_A : \Gamma.A \to \Gamma$, 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 $\mathcal{C}/\Gamma$. Therefore, any type should determine such an object.

This will be made precise in chapter 2 in the context of a CwF. However, to motivate the general definition, 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 $\mathcal{C}/\Gamma$, and investigate the structure and properties that this mapping should have.

1.4.4 *Terms*

Given a type A over the context Γ, a *term* a of type A is a morphism

$$a : \Gamma \to \Gamma.A$$

that is a section of the display map p_A, i.e. such that $p_A \circ a = \mathrm{id}$.

The idea of this definition is that a term of type A is defined to be exactly what is required to give an interpretation of the extended context $\Gamma.A$ 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 a is term of type A over the context Γ, we will write the judgement

$$\Gamma \vdash a : A$$

or simply $a : A$, when the context is clear.

For technical reasons, although terms can be regarded as a defined notion, we will take them as primitive in definition 2.1.1 below. Of course, the characterisation as sections is still valid, and will be proved as proposition 2.1.2.

1.4.5 *Substitutions*

Given a morphism $\sigma : \Delta \to \Gamma$, 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 A over Γ, there should exist a type $A[\sigma]$ over Δ, and a morphism $\sigma^+ : \Delta.A[\sigma] \to \Gamma.A$, which we refer to as σ *extended with* A.

The property of being able to transport terms of type A to terms of type $A[\sigma]$ can be expressed concisely by requiring that the following square

$$\begin{tikzcd} \Delta.f^*A \arrow[r, "\sigma^+"] \arrow[d, "p_{A[\sigma]}"'] & \Gamma.A \arrow[d, "p_A"] \\ \Delta \arrow[r, "\sigma"] & \Gamma \end{tikzcd} \tag{2}$$

be a pullback.

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

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

1.4.6 *Dependent products*

In order to define a notion of "function" internal to our system, we need to be able, given types A and B over some context Γ, to define a type $A \to B$, whose terms can be thought of as functions from A to B.

More generally, given a type A over Γ, and a type B over $\Gamma.A$, we want to define a type of *dependent functions* from A to B, the so called *dependent product* of A and B, which we denote by $\Pi_A B$.

Terms of $\Pi_A B$ can be thought of as functions whose result *type* depends on the argument. Alternatively, one can think of $\Pi_A B$ as an internalised form of the categorical product of a family of types.

We define dependent products rigorously in definition 2.1.20, but for now, we can think of $\Pi_A B$ as defined by the fact its terms are in natural bijective correspondence with terms of type B in the context $\Gamma.A$. This expresses the idea that a function is completely characterised by its value on a "generic" element of its domain.

1.4.7 *Dependent sums*

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

Given a type A over Γ, and a type B over $\Gamma.A$, the dependent sum of A and B, denote $\Sigma_A B$, intuitively represents the type of all pairs of terms a and b, where $a : A$ and $b : B[a]$. Dually to dependent products, dependent sums can be thought of as an internal version of the coproduct of a family of types.

Again, we will later give a precise definition of Σ (definition 2.1.22), but for now, we can think of $\Sigma_A B$ as a type characterised by the fact that its terms are in bijective correspondence with pairs of terms as above.

1.4.8 *Equality*

The final essential idea that we will require in order to replicate basic logic and set theoretical constructions in our system is that of *equality*.

The "structural" nature of the kind of system that we are set to create implies that we should only be allowed to consider equality between terms of the *same* type.

Given a type A in the context Γ, and terms $a, b : A$, we can then form an *equality type* $a = b$. This is the first point in our development where the type-theoretic incarnation of a concept differs substantially with its conventional set-theoretic counterpart.

In the usual classical foundations of mathematics (e.g. ZFC over some form of first-order logic), equality of sets is not itself a set, but a meta-theoretic entity. In other words, equality of mathematical objects is not itself a mathematical object.

One can of course remedy this somewhat by reifying equality into a set as follows: define the *equality set* $[a = b]$ of a and b as the equaliser [2]

$$\begin{tikzcd}[{[a = b]}] \arrow[r] & 1 \arrow[r, shift left, "a"] \arrow[r, shift right, "b"'] & X\end{tikzcd}$$

regarded as a subobject of a canonically specified terminal object 1 in Set (e.g. the ordinal 1).

This is indeed one way to interpret type theoretic equality in terms of sets, but the advantage (or the curse, depending on how one looks at it) of the type-theoretic account is that it is much more general, and the notion of equality set described above is but one of many possible interpretations.

We will define one version of equality precisely in definition 2.1.24, and another one later in section 2.4. Again, for this introductory discussion, we will limit ourselves to an informal characterisation: the equality type $a = b$ is defined by the following two features:

- a canonical term $\mathsf{refl} : a = a$;
- a "substitution" principle: a term $p : a = b$ can be used to reduce any construction involving b (and possibly p itself) into one in terms of a (and refl).

The first feature simply expresses the fact that every object should be equal to itself (and gives us a concrete *witness* of the fact). The second formalises the idea that equal objects are indistinguishable from within the theory.

1.4.9 *Propositions as types*

Equality as a type is but one example of a general pattern in type theory: propositions, i.e. statements *about* mathematical objects, are themselves mathematical objects and can be studied as such.

The idea is that if a type A is thought of as a proposition, then its terms are interpreted as *witnesses* of the truth of A, or, in other words, as pieces of evidence for A.

Interestingly, all the structures introduced above have a sensible interpretation in terms of operations over propositions. For example, if A and B are propositions, the type $\Pi_A B$ can be interpreted as the proposition stating that A

2 When working in a non-constructive meta-theory like ZF, the above definition can be simplified as follows: $[a, b]$ is defined to be 1 if $a = b$, and the empty set otherwise.

implies B: a witness of $\Pi_A B$, in fact, is a function that turns evidence for A into evidence for B.

Similarly $\Sigma_A B$ corresponds to the logical conjuction of A and B: a witness of $\Sigma_A B$ is a pair of witnesses for A and B respectively.

Furthermore, one can use dependent products and sums to reproduce the ideas of universal and existential quantification of logical theories. For example, if A is any type, and B is thought of as a family of propositions indexed over A (or, equivalently, a "predicate" over A), the dependent product $\Pi_A B$ corresponds to the assertion that B holds for all the elements of A (i.e. $\forall x : A, B$). Dually, $\Sigma_A B$ serves as the assertion that there exists an element of A for which B holds (i.e. $\exists x : A, B$).

1.4.10 *Other structures*

Unfortunately, the structures of dependent products, dependent sums and equality defined above, although very powerful and versatile, are often not enough to express certain mathematical ideas. Examples of constructions that are not covered by those basic operations are: induction, disjoint unions, logical negation, quotients, and others.

For this reason, type theories usually include extra structures designed to deal with those requirements. In the following, after giving precise definitions of the basic structures defined above, we will give a generic definition of *type former* (section 2.2), encompassing most of the type-theoretic structures that are encountered in the literature.

This will allow us to work in a type theory (or model thereof) where the set of type formers is arbitrary, and does not need to be specified in advance. That in turn will make some of our results very general, only subject to certain conditions on the type formers involved, which can then be verified separately and independently.

We will not discuss those extra type formers in detail. We will define some of them in section 2.4, but only give a brief explanation. We refer the interested reader to [36, Chapter 1].

1.5 HOMOTOPY TYPE THEORY

The equality type $x = y$ introduced in section 1.4.8 expresses the idea that the two elements x and y are "identified" in some sense, and they can be substituted for each other.

However, by itself, it has somewhat awkward features, which make it hard to use it effectively when formalising mathematics in type theory.

First of all, it is not well-behaved when it comes to describing equality of functions and equality of types. For example, we cannot derive the principle of *function extensionality*, stating that two functions are equal whenever they are equal at every point. Therefore, this principle is usually taken as an axiom in most incarnations of type theory.

Secondly, the following question may come quite naturally after reading the informal definition of section 1.4.8: is every witness of equality equal to refl?

A superficial reading of the substitution principle of equality (corresponding to the so-called J-eliminator, which we will introduce rigorously in section 2.4) would suggest this to be the case, since it says that proving a property of equality can be reduced to proving the corresponding property for refl.

A careful examination, however, reveals a fault in this straightforward argument: given arbitrary terms $a, b : A$, and $p : a = b$, we cannot internally express the property of p of being equal to refl, because p and refl have different types. If we restrict ourselves to terms $p : a = a$, then our premise is not general enough, and we are not allowed to use the substitution principle.

In fact, it turns out that the question cannot be answered internally: it is consistent to assume that there exist proofs of equality which are not themselves equal to refl [20]. This implies that equality cannot be simply thought of as a "mere" proposition, since it carries potentially non-trivial internal structure.

From here, one can either dismiss this limitation as a failure of the definition of equality, and address it by adding the missing component as an extra assumption (see (24)), or embrace it, and fully explore its consequences.

Both approaches are viable, and have been pursued with great success. The first makes it possible to encode most, if not all, of existing informal mathematics (at least, if we also assume certain classical principles such as the axiom of choice or the excluded middle). It is very close in spirit to working within the Mitchell-Bénabou language of topoi, and it exists on a similar level of generality. We will call such a theory *strict*.

The second approach is embodied by HoTT [36]. When no assumptions on the triviality of equality types is made, we can observe that types arrange themselves into a cumulative hierarchy of *truncation levels*, starting with -1-*types* (also called *propositions*), whose equality types are completely trivial, followed by 0-*types*, or *sets*, having propositions as equality types, and in general n-types, defined as those types whose equality types are $(n-1)$-types.

One appeal of HoTT is that equalities can be seen as paths in a space, and it is even possibly to develop substantial amounts of homotopy theory synthetically (see for example [9] for an extensive account). An important fact to keep in mind is that, when doing homotopy theory in type theory, every statement that one can make holds up to homotopy, and every construction respects (homotopy) equivalence.

This means that whatever we do will be "invariant", in the sense that it can only take the homotopy type of spaces, and homotopy equivalence classes of maps, into account, and not the concrete representations of spaces or maps. This is often considered a selling point of HoTT: one might perform constructions using representatives of homotopy classes in traditional homotopy theory, which make it necessary to show that those constructions are well-defined, i.e. do not depend on the choice of the representative.

In HoTT, everything is automatically well-defined up to homotopy as we are simply not able to talk about non-homotopy-invariant notions like strict equality internally.

1.6 THE PROBLEM OF "INFINITE STRUCTURES"

It is not hard to imagine that the blessing of having only constructions up to homotopy can turn out to be a curse: the inability to reflect a notion of "strict equality" into the theory can sometimes make certain ideas much harder to express.

For example, we cannot form a type expressing that a given diagram commutes strictly; all we can do is stating that it commutes up to homotopy. Unfortunately, depending on the shape of the diagram, this will only be sufficient in the simplest cases. More often than not, it will be necessary to say that the different "pieces" (the equalities expressing commutativity) fit together.

For instance, the fact that a certain sub-diagram commutes can be part of the proof that the diagram commutes, but it may at the same time be derivable as the composition of the fact that other sub-diagrams commute. In this case, it is natural to require these different ways of getting a certain proof to be equal. It does not stop here; these new proofs can themselves be required to be coherent, and so on.

This phenomenon is of course not something that can only be observed in type theory. The first step becomes already apparent in the theory of monoidal categories in the form of "Mac Lane's Pentagon". On higher dimensions, it is exactly the same issue that is discussed as *homotopy commutativity versus homotopy coherence* by Lurie [25].

In general, homotopy coherence corresponds to infinite towers of coherence data, and it is a major open problem (and commonly believed to be unsolvable) to express such towers internally in HoTT. One way to avoid the problem altogether is to restrict constructions to types of low truncation levels. As an example, the category theory developed in [2] only considers 1-truncated types to develop a theory of ordinary categories. This is in many situations not satisfactory: we know that types are ∞-groupoids [24, 37], and similarly, the universe should be an $(\infty, 1)$-category. Unfortunately, there does not seem be a way to express this internally in HoTT.

Of course, it is always possible to take one of the existing models of higher categories and replicate it internally in HoTT. However, since all of the existing models are ultimately built out of sets, this would force the HoTT version to be based on *sets* as well (i.e. 0-truncated types), which means that many specific structures that are expected to be $(\infty, 1)$-categories would not qualify. One notable example is provided by universes, which cannot in general be assumed to be truncated (as shown in [23]), hence cannot possibly be given a categorical structure for any notion of higher category which is based on sets. On the other hand, we define an $(\infty, 1)$-category structure for a universe in section 4.6.2.

The crucial shortcoming of HoTT is that we are unable to encode certain constructions which would appear to be harmless, as they only require finite amounts of coherence data at every step. An example that has received considerable attention in the HoTT community is the construction of Reedy fibrant n-semi-simplicial types (simply referred to as *semi-simplicial types*).

Let us start with Δ_+, the category of finite non-zero ordinals and strictly monotone functions. Let us write $[\mathsf{n}]$ for the ordinal with $n + 1$ elements. A type-valued diagram over Δ_+^{op} is a *strict* functor from Δ_+^{op} to the category of types. It would correspond to a type $X_{[\mathsf{n}]}$ (for simplicity written X_n) for every n, and face maps $d_i : X_{n+1} \to X_n$ for $0 \leq i \leq n$, as it is well-known that any map in Δ_+^{op} can be written as a composition of face maps. The problem is that we need the semi-simplicial identities (essentially a representation of the functor laws) to be strict, a fact which we cannot express in type theory.

The considered approach to avoid this problem is to only attempt internalising *Reedy fibrant* diagrams over Δ_+^{op}, essentially ensuring that the face maps are simple projections.

Using the correspondence between fibrations and type families, a (Reedy fibrant) semi-simplicial type then corresponds to a type X_0 (the "points") on level 0. On level 1, we need a family

$$X_1 : X_0 \to X_0 \to \mathcal{U},$$

where $\mathcal{U}$ is the universe of types. We think of X_1 as lines between types. Next, we need

$$X_2 : \Pi_{a,b,c:X_0} X_1(a, b) \to X_1(b, c) \to X_1(a, c) \to \mathcal{U},$$

the type of fillers for triangles.

Writing down the type of X_4 is already rather tedious, but nevertheless straightforward: X_4 is a family which gives a type for any collection of four points, six lines and four triangles that form a boundary of a tetrahedron.

A long-standing open problem of homotopy type theory is then to write down the type of X_n, or something equivalent to it, for a general natural number n.

This has revealed to be much harder than one might expect, and it is actually conjectured to be impossible.

What is definitely possible is to generate an expression X_n for every externally fixed numeral n, such that the expressions $X_0, X_1, X_2, \dots$ all "fit together". If one attempts to perform the same construction for a *variable* $n : \mathbb{N}$, the types do not match up anymore. The reason is that some strict equalities that hold in the case of a numeral n fail to hold in the case of a variable. One could try to prove that the required equalities hold up to homotopy, but one quickly realises that one would also need to show that these equalities are coherent, and that the coherence proofs are coherent themselves, and so on; even only expressing the coherence data that is required to make the construction go through seems to be as hard, if not harder, than the original problem.

1.7 INTERNALISING STRICT EQUALITY

In some sense, the equalities needed when attempting to construct semi-simplicial types, as explained in section 1.6, *should* hold and be fully coherent, because they are trivially satisfied for each externally fixed natural number. If only we had a way to reason about strict equalities *within* the system, there would be no problem at all; however, this would require strict equalities to be reified into a type.

We could take the equality of a strict theory to be the internalised version of strict equality. In that case, it would be possible to construct Reedy fibrant semi-simplicial types internally. However, we can also simply define categories and functors in the usual sense, and all coherences will be satisfied automatically thanks to the strictness assumptions in the theory.

Using this approach, we would bypass all the coherence problems, but have to give up all the advantages of HoTT, like univalence and higher inductive types. The idea of a two-level system is to combine strict type theory and HoTT, instead of viewing them as two alternative extensions of the basic underlying type theory.

A two-level type theory consists of two "parallel" type theory, with possibly different structures, sharing a small common *core* consisting of dependent products and sums. We call the two fragments *strict* and *fibrant* respectively. The strict fragment is, unsurprisingly, a strict form of type theory, while the fibrant fragment is an incarnation of HoTT. Every fibrant type can be canonically regarded as a strict type, but not vice versa.

The reason why two-level type theory has to be set up in this way, rather than just having two equality types, is lemma 4.2.1, showing that if there is no distinction between fibrant and strict types, then the two equalities necessarily collapse into one.

The idea a type theory with two equality types is not new. Such a system was first suggested by Voevodsky [38], who referred to it as HTS, but the theory developed in this thesis (specifically in chapter 3) presents substantial differences with HTS (see section 4.1.1). In particular, it requires no form of equality reflection in its strict fragment. Thus, we can avoid all the problems that are usually connected to equality reflection, such as undecidability of type checking.

In contrast, the two-level system presented in this thesis is well-behaved, very close to the standard formulation of HoTT, and has straightforward semantics. One could expect that a downside of our system might be reduced expressibility compared to a theory that features equality reflection. However, we can achieve in our system what HTS was suggested for: a definition of semi-simplicial types, and other constructions based on them.

Furthermore, by being careful about the relationship between strict and fibrant type formers, we can prove a conservativity result (theorem 3.2.14). This means that, in some sense, the fibrant fragment corresponds exactly to HoTT as presented in [36]. In a proof assistant which supports this theory, we could in principle implement results that so far can only be stated meta-theoretically. To give an example, it is shown in [22] that constant functions from A to B which satisfy n coherence conditions correspond to maps $\|A\| \to B$, provided that B is n-truncated. Here n is a natural number, external to the theory, so the result has to be formalised as a *sequence* of internal statements, which means that it can only be stated and proved meta-theoretically. In a two-level system, we can formalise it by taking n to be an element of the strict type of natural numbers, then show the required equivalence in the fibrant fragment. Conservativity would then allow us to conclude the the corresponding statement is valid in HoTT for all choices of the parameter n, and all the complications of meta-theoretic reasoning would be encapsulated in the proof of theorem 3.2.14.

2

TYPE THEORY AND TYPE FORMERS

This chapter contains the fundamental definitions and constructions that will be used throughout the rest of the thesis. We will start from the intuitive ideas presented in chapter 1, and make them precise in terms of *categories with families*, which we choose as the primary basic notion of model of type theory.

Our presentation of basic type formers (Π, Σ, equality and unit type) is based on the same ideas as in [5], which will make it easier to extend the notion of type former to more general operations, as well as to the context of chapter 3.

2.1 CATEGORIES WITH FAMILIES

Definition 2.1.1 (see [11]). A *category with families* (CwF) is given by:

- a category $\mathcal{C}$, equipped with a distinguished terminal object 1;
- a presheaf $\mathrm{Ty} : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$;
- a presheaf $\mathrm{Tm} : (\int \mathrm{Ty}) \to \mathsf{Set}^{\mathrm{op}}$;
- for all $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, an object $(\Gamma.A, \pi_A)$ representing the functor $(\mathcal{C}/\Gamma) \to \mathsf{Set}^{\mathrm{op}}$ defined by:

$$(\Delta, \sigma) \mapsto \mathrm{Tm}_\Delta(A[\sigma]). \tag{3}$$

Here and in the following, if $X : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$ is a presheaf on a category $\mathcal{C}$, $\sigma : \mathcal{C}(\Delta, \Gamma)$ is a morphism, and $x : X_\Gamma$ is an element of X, we write $x[\sigma]$ instead of $X(\sigma)(x)$.

The objects of $\mathcal{C}$ are called *contexts*. Given a context Γ, the elements of $\mathrm{Ty}(\Gamma)$ are called *types*, and given a type A, the elements of $\mathrm{Tm}_\Gamma(A)$ are called *terms*.

The context $\Gamma.A$ is called the *context extension* of Γ by the type A, and π_A is the *display map* of A.

The action of Ty and Tm on morphisms is called *substitution*.

Note that, given a morphism $\sigma : \mathcal{C}(\Delta, \Gamma)$, a type $A : \mathrm{Ty}(\Gamma)$, and a term $a : \mathrm{Tm}_\Delta(A[\sigma])$, the definition of CwF gives a corresponding morphism $\mathcal{C}(\Delta, \Gamma.A)$ which we will denote by $\langle \sigma, a \rangle$.

Proposition 2.1.2. *For all contexts* $\Gamma : \mathcal{C}$ *and types* $A : \mathrm{Ty}(\Gamma)$*, there is a natural isomorphism:*

$$\mathrm{Tm}_\Gamma(A) \cong \mathcal{C}/\Gamma(\Gamma, \Gamma.A). \tag{4}$$

Proof. Equation (4) follows directly from the definition of context extension, by taking $\Delta :\equiv \Gamma$ and $\sigma :\equiv \mathrm{id}$ in (3). □

Proposition 2.1.2 says that terms of type A can be equivalently regarded as sections of the display map $\pi_A : \mathcal{C}(\Gamma.A, \Gamma)$.

Proposition 2.1.3. *Let* $\sigma : \mathcal{C}(\Delta, \Gamma)$ *be any morphism, and* $A : \mathrm{Ty}(\Gamma)$*. There exists a morphism* $\sigma^+ : \mathcal{C}(\Delta.A[\sigma], \Gamma.A)$ *that makes the square*

$$\begin{tikzcd} \Delta.A[\sigma] \arrow[r, "\sigma^+"] \arrow[d, "\pi_{A[\sigma]}"'] & \Gamma.A \arrow[d, "\pi_A"] \\ \Delta \arrow[r, "\sigma"'] & \Gamma \end{tikzcd} \tag{5}$$

into a pullback.

Proof. Diagram 5 being a pullback is equivalent to the condition that, for all contexts Φ and morphisms $\tau : \mathcal{C}(\Phi, \Delta)$, there is a natural isomorphism:

$$\mathcal{C}/\Gamma(\sigma_* \Phi, \Gamma.A) \cong \mathcal{C}/\Delta(\Phi, \Delta.A[\sigma]),$$

where we write Φ to mean the pair (Φ, τ) in the slice category $\mathcal{C}/\Delta$, and similarly for $\Gamma.A$ and $\Delta.A[\sigma]$.

But clearly, the isomorphism holds, since both sides are naturally isomorphic to $\mathrm{Tm}_\Phi(A[\sigma \circ \tau])$, by the defining property of context extension. □

Proposition 2.1.3 allows us to turn the context extension operation into a functor $\mathsf{ext} : \int \mathrm{Ty} \to \mathcal{C}$.

Definition 2.1.4. Let $\mathcal{C}$, $\mathcal{D}$ be CwFs. A *CwF morphism* $\mathcal{C} \to \mathcal{D}$ is given by:

- a functor $F : \mathcal{C} \to \mathcal{D}$;
- a natural transformation $F^{\mathrm{Ty}} : \int_\Gamma \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(F\Gamma)$;
- a natural transformation $F^{\mathrm{Tm}} : \int_{\Gamma, A} \mathrm{Tm}_\Gamma(A) \to \mathrm{Tm}_{F\Gamma}(F^{\mathrm{Ty}} A)$;

such that $F1$ is a terminal object in $\mathcal{D}$, and, for all $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, the map

$$\phi^F_A : \mathcal{D}(F(\Gamma.A), F\Gamma.F^{\mathrm{Ty}} A)$$

defined below is an isomorphism.

The map ϕ_A^F is obtained as follows. First, by applying the functor F to the display map p_A, the context $F(\Gamma.A)$ can be regarded as an element of $\mathcal{D}/F\Gamma$. Then, the term $F^{\mathrm{Tm}}(v_A) : \mathrm{Tm}_{F(\Gamma.A)}(F^{\mathrm{Ty}}A[F(p_A)])$ determines a morphism $\mathcal{D}/F\Gamma(F(\Gamma.A), F\Gamma.F^{\mathrm{Ty}}A)$ by the defining property of context extension, and ϕ_A^F is taken to be the corresponding underlying morphism $\mathcal{D}(F(\Gamma.A), F\Gamma.F^{\mathrm{Ty}}A)$.

We will usually omit the superscripts Ty and Tm when referring to the action of a morphism on types and terms respectively.

Definition 2.1.5. A CwF morphism $F : \mathcal{C} \to \mathcal{D}$ is said to be *split* if it preserves the distinguished terminal objects and context extension "on the nose" and the map ϕ_A^F is the identity for all types A.

Definition 2.1.6. A CwF morphism $F : \mathcal{C} \to \mathcal{D}$ is said to be a *CwF equivalence* if it is an equivalence of categories, and it induces isomorphisms on types.

Note that a CwF equivalence automatically induces isomorphisms on terms.

2.1.1 *Notation*

In the following, let $\mathcal{C}$ be a CwF.

If Γ is a context, and $A : \mathrm{Ty}(\Gamma)$, the universal property of the context extension, applied to the identity substitution $\mathcal{C}/\Gamma(\Gamma.A, \Gamma.A)$, yields a canonical term $v_A : \mathrm{Tm}_{\Gamma.A}(A[\pi_A])$. We call v_A the *variable* of type A.

Weakenings, i.e. substitutions along display maps, will often be omitted from the notation, as they can usually be unambiguously reconstructed, and leaving them implicit simplifies the syntax considerably. In particular, the variable of type A can be regarded simply as a term in $\mathrm{Tm}_{\Gamma.A}(A)$.

Sometimes, when building contexts using context extension, we will associate "names" to certain types. These names will be used to refer to their corresponding variables, and weakenings thereof. For example, the context $\Gamma(a : A)$ denotes the context $\Gamma.A$, with the convention that the name a refers to the variable $v_A : \mathrm{Tm}_{\Gamma(a:A)}(A)$.

The terminal object of $\mathcal{C}$ is referred to as the *unit* context.[1] We will identify types in the unit context with the corresponding contexts obtained by context extension. So, for example, if $A : \mathrm{Ty}(1)$, we will write A to denote $1.A$, and if $B : \mathrm{Ty}(A)$, we can form the context extension $A.B$.

1 In traditional type-theoretic terminology, the term *empty context* is more often found. This is because contexts are usually built explictly by chaining a finite number of context extensions, and 1 is the base case of this process, where no extensions have been performed yet. However, "empty" is more suggestive of an initial, rather than terminal, object, so we will keep consistency with the corresponding terminology for types, and use the term *unit context* instead

Finally, thanks to proposition 2.1.2, terms in $\mathrm{Tm}_\Gamma(A)$ correspond bijectively with sections of the display map π_A. We will therefore identify a term with its corresponding section.

With those syntactical conventions, working in an arbitrary CwF is basically indistinguishable from working in the corresponding type theory (i.e. its internal language). For that reason, we are able to avoid giving a precise definition of *syntax* of type theory. Our definitions and constructions exist purely within the semantics realm of CwFs, and that is sufficient for our purposes.

We will also implicitly assume the existence of a hierarchy of an arbitrary finite number of universes of sets $\mathsf{Set}_0 \subseteq \mathsf{Set}_1 \subseteq \mathsf{Set}_2 \ldots$, but remove the indices from the notation. In particular, we will simply write Set instead of Set_0 or Set_1. This is in line with a widespread convention in type theory called "typical ambiguity" [12], and is used, for example, in [36].

The existence of this hierarchy of universes may depend on certain large cardinal axioms (like the existence of a corresponding chain of innaccessible cardinals) in a foundations like ZFC. Alternatively, if we assume that the metatheory that we are working in is itself some form of type theory, then all we need is a tower of universes (as in definition 2.1.15) in the outer theory.

2.1.2 *Presheaves*

The prototypical example of a CwF is the category of presheaves over $\mathcal{C}$, where $\mathcal{C}$ is an arbitrary (small) category. We will denote this category by $\widehat{\mathcal{C}}$. For any presheaf P, let $\mathbf{Ty}(P)$ be the category of presheaves over $\int^{\mathcal{C}} P$, and let $\mathrm{Ty}(P)$ be the underlying set of objects of $\mathbf{Ty}(P)$.

Clearly, $\mathbf{Ty}$ defines a functor $\mathcal{C}^{\mathrm{op}} \to \mathsf{Cat}$, hence Ty is a functor $\mathcal{C}^{\mathrm{op}} \to \mathsf{Set}$. The corresponding term functor is given by:

$$\mathrm{Tm}_P(A) :\equiv \mathbf{Ty}(P)(1, A),$$

where 1 is the terminal object of $\mathbf{Ty}(P)$, i.e. the functor which is constantly equal to the terminal object 1 of Set. Substitutions are defined in the obvious way via precomposition.

To define context extension, we will need the following

Proposition 2.1.7. *Let $\mathcal{C}$ be any category, and $P : \widehat{C}$ a presheaf on $\mathcal{C}$. There is an equivalence of categories:*

$$\mathbf{\Phi} : \widehat{C}/P \cong \widehat{\int P}$$

such that, for all presheaves Q over P, there is an isomorphism of categories:

$$\int \mathbf{\Phi}(Q) \cong \int Q \tag{6}$$

Proof. Given a presheaf Q over P, define a presheaf $\mathbf{\Phi}(Q)$ on $\int^{\mathcal{C}} P$ by assigning to every object (Γ, x) of $\int^{C} P$, where $\Gamma : \mathcal{C}$ and $x : P_\Gamma$, the fibre of Q over x.

Conversely, given a presheaf $F : \widehat{\int^{C} P}$, define Q_Γ as the set of pairs (x, y), where $x : P_\Gamma$, and $y : F(\Gamma, x)$.

It is easy to see that $\mathbf{\Phi}$. defines an equivalence of categories. As for equation (6), it follows immediately from the definition of $\mathbf{\Phi}$. □

Now, given a presheaf P and a type A over P, define $P.A$ to be the presheaf over P corresponding to A through the equivalence of proposition 2.1.7, so that we have equivalences:

$$\mathbf{Ty}(P.A) \cong \widehat{\int A} \cong \mathbf{Ty}(P)/A, \tag{7}$$

where the first is a consequence of the isomorphism (6), and the second is obtained by applying proposition 2.1.7 to the category $\int^{C} P$. We will call $P.A$ the *total space* of A.

Therefore, we can associate, to any type in $B : \mathrm{Ty}(P.A)$, a corresponding type in $\mathrm{Ty}(P)$, which we will denote by $\Sigma_A B$. Note that $P.\Sigma_A B \cong P.A.B$.

Lemma 2.1.8. *The map $B \mapsto \Sigma_A B$ defines a left adjoint for the substitution functor $\mathbf{Ty}(P) \to \mathbf{Ty}(P.A)$ along π_A.*

Proof. The functor Σ_A can be regarded as the composition:

$$\Sigma_A : \mathbf{Ty}(P.A) \to \mathbf{Ty}(P)/A \to \mathbf{Ty}(P),$$

where the first functor is the equivalence 7, and the second is the forgetful functor.

The latter has a right adjoint, mapping a type $C : \mathrm{Ty}(P)$ to the product $A \times C$, together with the first projection.

Therefore, all is left to do is to verify that $A \times C$ corresponds to $C[\pi_A]$ through the equivalence 7, which is easy to see. □

Note that $\mathbf{Ty}(P)$, being a presheaf category, is a cartesian closed category with all small limits and colimits. In particular, given two types A, B, we can form their exponential B^A, which we can think of as the "function type" between A and B.

We will now generalise this notion of function type to the situation where B "depends on A", i.e. when B is not in $\mathbf{Ty}(P)$, but in $\mathbf{Ty}(P.A)$.

Given $B : \mathrm{Ty}(P.A)$, we can obtain a type $\Sigma_A B : \mathrm{Ty}(P)$, together with a projection $\pi_1 : \mathrm{Ty}(P)(\Sigma_A B, A)$. Since $\mathrm{Ty}(P)$ has limits, we can form a pullback square:

$$\begin{tikzcd}
\Pi_A B \arrow[r] \arrow[d] & (\Sigma_A B)^A \arrow[d] \\
1 \arrow[r] & A^A,
\end{tikzcd} \tag{8}$$

where the bottom arrow selects the identity morphism $A \to A$.

This determines a type $\Pi_A B : \mathrm{Ty}(P)$.

Lemma 2.1.9. *The map $B \mapsto \Pi_A B$ defines a right adjoint for the substitution functor $\mathbf{Ty}(P) \to \mathbf{Ty}(P.A)$ along π_A.*

Proof. Let X be an arbitrary type in $\mathrm{Ty}(P)$, and consider the homset $\mathbf{Ty}(P.A)(X[\pi_A], B)$. Through the equivalence (7), this is isomorphic to $(\mathbf{Ty}(P)/A)\,(A \times X, \Sigma_A B)$, which fits into a pullback square:

$$\begin{tikzcd}
(\mathbf{Ty}(P)/A)\,(A \times X, \Sigma_A B) \arrow[r] \arrow[d] & \mathbf{Ty}(P)(A \times X, \Sigma_A B) \arrow[d] \\
1 \arrow[r] & \mathbf{Ty}(P)(A \times X, A).
\end{tikzcd}$$

Using the adjunction defining the exponential, this diagram is isomorphic to:

$$\begin{tikzcd}
(\mathbf{Ty}(P)/A)\,(A \times X, \Sigma_A B) \arrow[r] \arrow[d] & \mathbf{Ty}(P)(X, (\Sigma_A B)^A) \arrow[d] \\
1 \arrow[r] & \mathbf{Ty}(P)(X, A^A).
\end{tikzcd}$$

However, by applying the limit-preserving functor $\mathbf{Ty}(P)(X, -)$ to 8, we get the same diagram, but with $\mathbf{Ty}(P)(X, \Pi_A B)$ in the top left corner. Therefore, it follows that there is a natural isomorphism

$$\mathbf{Ty}(P)(X, \Pi_A B) \cong \mathbf{Ty}(P.A)(X[\pi_A], B),$$

hence Π_A is right adjoint to substitution along π_A. □

As an immediate consequence of lemma 2.1.9, there is a natural isomorphism:

$$\lambda : \mathrm{Tm}_{P.A}(B) \to \mathrm{Tm}_P(\Pi_A B), \tag{9}$$

which is often referred to as *lambda abstraction*. Furthermore, given terms $f : \mathrm{Tm}_P(\Pi_A B)$ and $a : \mathrm{Tm}_P(A)$, we get a term $\lambda^{-1}(f)[a] : \mathrm{Tm}_P(B[a])$. It is customary to denote this term simply by $f\ a$, and call this operation *application*.

Alternatively, we can regard application as a morphism $\epsilon_{A,B}$:

$$\begin{tikzcd}
\Gamma.A.\Pi_A B \arrow[rr, "\epsilon_{A,B}"] \arrow[dr] & & \Gamma.A.B \arrow[dl] \\
& \Gamma.A. &
\end{tikzcd}$$

Since the type B appearing in a $\Pi_A B$ is defined over an extended context, it is often convenient to introduce a name for the variable of type A, when constructing such an expression. Therefore, we will employ the notation:

$$\Pi_{a:A} B,$$

to mean the exact same thing as $\Pi_A B$, with the addition that B is assumed to be a type in the context $P(a : A)$, i.e. the name a refers to the variable of type A within the expression that defines B. A similar notation will be used for Σ.

We will now define a very simple notion of *equality type* for presheaves.

Let P be a presheaf, and $A : \mathrm{Ty}(P)$ a type over it. Consider the diagonal morphism $\mathbf{Ty}(P)(A, A \times A)$ and map it through the equivalence of proposition 2.1.7 to get a morphism in $\widehat{C}(P.A, P.(A \times A))$, which is isomorphic to $\widehat{C}(P.A, P.A.A)$. Using proposition 2.1.7 again, this morphism determines a type over $P.A.A$ which we will denote by Eq_A, and refer to as the *equality type* of A.

In particular, given terms $a_1, a_2 : \mathrm{Tm}_P(A)$, we can form a type $\mathsf{Eq}_A[a_1, a_2]$ by substitution. Terms of this type are witnesses of equality betwee a_1 and a_2, hence this type is inhabited (i.e. it has a global section) if and only if a_1 and a_2 are equal terms.

Lemma 2.1.10. *The type* Eq_A *is a subterminal object of* $\mathbf{Ty}(P.A.A)$.

Proof. Since equivalence of categories preserves subterminality, it is enough to show that the diagonal $A \to A \times A$ is subterminal in $\mathbf{Ty}(P)/(A \times A)$.

Let now $\mathcal{C}$ be any category, and $A : \mathcal{C}$ an object such that the product $A \times A$ exists. The diagonal $\delta : A \to A \times A$ is the equaliser of the two projections $A \times A \to A$, hence it is monic. Since the forgetful functor $\mathcal{C}/(A \times A) \to \mathcal{C}$ is faithful, it follows that $\delta \to \mathrm{id}$ is monic in $\mathcal{C}/(A \times A)$, i.e. δ is subterminal. □

2.1.3 *Basic type formers*

In the previous section, we defined the operations Σ, Π and Eq on types of a presheaf category. We will now define what it means for a general CwF to support those operations.

The following definitions are standard (see for example [18]).

Definition 2.1.11. We say that a CwF *supports* Π*-types* if for any two types $A : \mathrm{Ty}(\Gamma)$ and $B : \mathrm{Ty}(\Gamma.A)$ there is a type $\pi(A, B) : \mathrm{Ty}(\Gamma)$, and for each $b : \mathrm{Tm}_{\Gamma.A}(B)$ there is a term $\lambda(b)$, and for each $f : \mathrm{Tm}_\Gamma(\pi(A, B))$ and $a : \mathrm{Tm}_\Gamma(A)$ there is a term $f \cdot a : \mathrm{Tm}_\Gamma(B[a])$ such that the following equations (appropriately quantified) hold:

$$\begin{aligned}
&\lambda(b) \cdot a = b[a] \\
&\lambda(f \cdot v_A) = f \\
&\pi(A, B)[\tau] = \pi(A[\tau], B[\tau^+]) \\
&(\lambda(b))[\tau] = \lambda(b[\tau]) \\
&(f \cdot a)[\tau] = f[\tau] \cdot a[\tau].
\end{aligned}$$

Definition 2.1.12. We say that a CwF *supports* Π*-types* if for any two types $A : \mathrm{Ty}(\Gamma)$ and $B : \mathrm{Ty}(\Gamma.A)$ there is a type $\sigma(A, B) : \mathrm{Ty}(\Gamma)$, and for each $a : \mathrm{Tm}_\Gamma(A)$ and $b : \mathrm{Tm}_\Gamma(B[a])$ there is a term $\langle a, b\rangle : \mathrm{Tm}_\Gamma(\sigma(A, B))$, and for all terms $x : \mathrm{Tm}_\Gamma(\sigma(A, B))$ there are terms $\pi(x) : \mathrm{Tm}_\Gamma(A)$ and $\pi'(x) : \mathrm{Tm}_\Gamma(B[\pi(x)])$ such that the following equations (appropriately quantified) hold:

$$\begin{aligned}
&\pi(\langle a, b\rangle) = a \\
&\pi'(\langle a, b\rangle) = b \\
&\langle \pi(x), \pi'(x)\rangle = x \\
&\sigma(A, B)[\tau] = \sigma(A[\tau], B[\tau^+]) \\
&\langle a, b\rangle[\tau] = \langle a[\tau], b[\tau]\rangle \\
&\pi(x)[\tau] = \pi(x[\tau]) \\
&\pi'(x)[\tau] = \pi(x'[\tau]).
\end{aligned}$$

Definition 2.1.13. We say that a CwF *supports equality types* if for all types $A : \mathrm{Ty}(\Gamma)$ there is a type $\mathsf{eq}(A) : \mathrm{Ty}(\Gamma.A.A)$, such that two terms $a, b : \mathrm{Tm}_\Gamma(A)$ are equal if and only if there is a term $p : \mathrm{Tm}_\Gamma(\mathsf{eq}(A)[a, b])$, and furthermore:

$$\mathsf{eq}(A)[\tau^{++}] = \mathsf{eq}(A[\tau]).$$

Definition 2.1.14. We say that a CwF *has a unit type* if there exists a type $1 : \mathrm{Ty}(1)$ with a unique term.

The purpose of this section is to develop equivalent formulations of the above definitions based on presheaves. In section 2.2, we will introduce the *rule framework*, and that will help us generalise the presheaf-based definitions (definition 2.1.20, definition 2.1.22, definition 2.1.24 and definition 2.1.26) to cover a wide variety of "type formers".

Definition 2.1.15. Let $\mathcal{C}$ be a CwF. A *universe* in $\mathcal{C}$ is given by:

- a type $\mathcal{U}$ in the unit context;
- a type El in the context $\mathcal{U}$.

We will see later how universes of sets determine universes in presheaf categories for an arbitrary $\mathcal{C}$ (section 2.1.6). For now, we will focus on the case where $\mathcal{C}$ is itself a CwF. In that case, the presheaf category $\widehat{\mathcal{C}}$ has a canonical universe, given by the functors Ty and Tm, part of the CwF structure of $\mathcal{C}$. For reasons that will be clear later, we will call this the *fibrant universe* of $\widehat{\mathcal{C}}$.

Since now we have two CwFs in play, in an attempt to avoid confusion, we will use the notation $\widehat{\mathrm{Ty}}$ and $\widehat{\mathrm{Tm}}$ when discussing the CwF structure on $\widehat{\mathcal{C}}$.

In the following, we will write y for the Yoneda embedding $\mathcal{C} \to \widehat{\mathcal{C}}$.

Lemma 2.1.16. *Let P be a presheaf on $\mathcal{C}$, A a term of type* Ty *in the context P of $\widehat{\mathcal{C}}$, and x an element of P over some $\Gamma : \mathcal{C}$. Let us write π for the display map of the type* $\mathrm{Tm}[A]$ *over P.*

There is an isomorphism of types over $P.\mathrm{Tm}[A]$:

$$y(\Gamma, x)[\pi] \cong y(\Gamma.A_\Gamma(x), x[\pi], v_{A_\Gamma(x)}), \tag{10}$$

natural in $(\Gamma, x) : \int P$.

Proof. We will construct the required isomorphism by using proposition 2.1.7 to transport all the presheaves involved to $\widehat{\mathcal{C}}$.

By the Yoneda lemma, we can regard x as a morphism $y(\Gamma) \to P$. The left side of 10 is then isomorphic to the type over $y(\Gamma)$ obtained by substituting $\mathrm{Tm}[A]$ along x.

As for the right side, its total space can also be regarded as a presheaf over $y(\Gamma)$ through the Yoneda embedding of the display map $\Gamma.A(x) \to \Gamma$.

By proposition 2.1.7, presheaves over $y(\Gamma)$ correspond to presheaves on $\int y(\Gamma)$, which is isomorphic to $\mathcal{C}/\Gamma$. Applying the isomorphism of proposition 2.1.7 explicitly, it is easy to see that the left side is mapped to the functor given by (3) for the type $A(x)$, so the conclusion follows from the defining property of context extension. □

Corollary 2.1.17. *Let P be a presheaf on $\mathcal{C}$, and A a term of type* Ty *in the context P of $\widehat{\mathcal{C}}$. The type*

$$\Pi_{\mathrm{Tm}[A]}\mathrm{Ty}$$

is isomorphic to the presheaf on $\int P$ given by:

$$(\Gamma, x) \mapsto \mathrm{Ty}(\Gamma.A_\Gamma(x)). \tag{11}$$

Proof. Again, let us write π for the display map of $\mathrm{Tm}[A]$.

Fix an arbitrary $(\Gamma, x) : \int P$. By lemma 2.1.9, there is a natural isomorphism:

$$\widehat{\mathbf{Ty}}(P)(y(\Gamma, x), \Pi_{\mathrm{Tm}[A]}\mathrm{Ty}) \cong \widehat{\mathbf{Ty}}(P.\mathrm{Tm}[A])(y(\Gamma, x)[\pi], \mathrm{Ty}).$$

By lemma 2.1.16, the weakened type $y(\Gamma, x)[\pi]$ is isomorphic to the representable presheaf $y(\Gamma, x, A(x))$, hence the conclusion follows from the Yoneda lemma. □

In the setting of 2.1.17, if B is a term of type $\Pi_{\mathrm{Tm}[A]}\mathrm{Ty}$ in context P, we will denote by $\widetilde{B}_\Gamma(x)$ the element of $\mathrm{Ty}(\Gamma.A_\Gamma(x))$ corresponding to $B_\Gamma(x)$ through the isomorphism 11. Expanding the definition of the isomorphism, one can show that:

$$\widetilde{B}_\Gamma(x) = (\lambda^{-1}B)_{\Gamma.A_\Gamma(x)}(x[\pi], v_{A_\Gamma(x)}).$$

Corollary 2.1.18. *Let P be a presheaf on $\mathcal{C}$, A a term of type* Ty*, and B a term of type $\Pi_{\mathrm{Tm}[A]}\mathrm{Ty}$, both in the context P. The type*

$$\Pi_{\mathrm{Tm}[A]}\mathrm{Tm}[B\ a]$$

is isomorphic to the presheaf on $\int P$ given by:

$$(\Gamma, x) \mapsto \mathrm{Tm}_{\Gamma.A_\Gamma(x)}(\widetilde{B}_\Gamma(x))$$

The universe Ty allows us to use the CwF structure on $\widehat{\mathcal{C}}$ to give definitions that work across all types of $\mathcal{C}$. However, to generalise Π and Σ, we need to access *pairs* of dependent types. For that reason, we define the context $\mathrm{Ty}^{(2)}$ as:

$$(\mathbf{A} : \mathrm{Ty})(\mathbf{B} : \Pi_{\mathrm{Tm}[\mathbf{A}]}\mathrm{Ty}).$$

Here we are using the syntactical conventions introduced in section 2.1.1. Let us take a minute to explain in detail what this expression means.

First of all, since Ty is a type in the unit context of $\widehat{C}$, we can form a context $P_0 :\equiv (\mathbf{A} : \mathrm{Ty})$ by extension from the unit context, and use $\mathbf{A}$ to refer to the corresponding term of type Ty, i.e. $\mathbf{A} : \widehat{\mathrm{Tm}}_{P_0}(\mathrm{Ty})$.

In the context P_0, the morphism corresponding to the variable $\mathbf{A}$ is just the identity $P_0 \to P_0$, hence $\mathrm{Tm}[\mathbf{A}]$ could have simply been written as Tm. However, using an explicit substitution makes it clear that we are referring to the variable $\mathbf{A}$, and generalises better to situations where the context contains more than one variable.

Since $T :\equiv \Pi_{\mathrm{Tm}[\mathbf{A}]}\mathrm{Ty}$ is a type in the context P_0, we can perform another context extension and obtain the context $P_0(\mathbf{B} : T)$. If we make weakenings explicit, now $\mathbf{A}$ refers to the variable of type $\mathrm{Ty}[\pi_{\mathrm{Ty}}][\pi_T]$, and $\mathbf{B}$ to the variable of type $T[\pi_T]$.

Corollary 2.1.19. *There is an isomorphism, natural in $\Gamma : \mathcal{C}$:*

$$\mathrm{Ty}^{(2)}(\Gamma) \cong \coprod_{A:\mathrm{Ty}(\Gamma)} \mathrm{Ty}(\Gamma.A).$$

Proof. Immediate consequence of corollary 2.1.17 and the definition of context extension of presheaves. □

Thanks to corollary 2.1.19, we are free to identify elements of $\mathrm{Ty}^{(2)}(\Gamma)$ with pairs of types (A, B), where $A : \mathrm{Ty}(\Gamma)$ and $B : \mathrm{Ty}(\Gamma.A)$. However, using $\mathrm{Ty}^{(2)}$ can sometimes be preferable, since it avoids referring to context extension at all.

Definition 2.1.20. A Π-type structure on $\mathcal{C}$ is given by:

- a term
$$\pi : \widehat{\mathrm{Tm}}_{\mathrm{Ty}^{(2)}}(\mathrm{Ty}), \tag{12}$$
- an isomorphism
$$\mathrm{Tm}[\pi] \cong \Pi_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a] \tag{13}$$
of types over $\mathrm{Ty}^{(2)}$.

Note that a Π-type structure on $\mathcal{C}$ is given entirely in terms of the CwF structure on $\widehat{\mathcal{C}}$ and its fibrant universe.

Definition 2.1.20 can be stated more explictly: giving the term 12 is the same as giving a natural transformation $\pi : \mathrm{Ty}^{(2)} \to \mathrm{Ty}$, and, thanks to corollary 2.1.18, the isomorphism 13 is equivalent to an isomorphism:

$$\mathrm{Tm}_\Gamma(\pi_\Gamma(A, B)) \cong \mathrm{Tm}_{\Gamma.A}(B). \tag{14}$$

It is then easy to verify that $\mathcal{C}$ supports Π-types (definition 2.1.11) if and only if it has a Π-type structure. In particular, we get the following:

Proposition 2.1.21. *For any category $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is equipped with a canonical Π-type structure.*

Proof. It looks like one could simply take π to be the Π operation on presheaves. However, Π, regarded as a family of functions $\widehat{\mathrm{Ty}}^{(2)}(\Gamma) \to \widehat{\mathrm{Ty}}(\Gamma)$, is not natural in Γ.

In fact, keeping in mind that $\widehat{\mathrm{Ty}}(\Gamma)$ is a *category*, and not just a set, one would only be able to prove that Π is a *pseudonatural* transformation of functors $\mathcal{C} \to \mathsf{Cat}^{\mathrm{op}}$. Fortunately, there is a way to give an alternative equivalent definition of Π that is indeed strictly natural.

Let $P : \widehat{\mathcal{C}}$, $A : \widehat{\mathrm{Ty}}(P)$, and $B : \widehat{\mathrm{Ty}}(P.A)$. We will define $\pi(A, B)$ as a functor $\int P \to \mathsf{Set}^{\mathrm{op}}$. For $(\Gamma, x) : \int P$, we will write $x : y(\Gamma) \to P$ for the morphism corresponding to x through the isomorphism of the Yoneda lemma. Then set:

$$\pi(A, B)_\Gamma(x) :\equiv \left(\Pi_{A[x]}B[x^+]\right)_\Gamma(\mathrm{id}).$$

Pseudonaturality of Π implies that $\pi(A, B) \cong \Pi_A B$. Furthermore, it is easy to check directly that $\pi : \mathrm{Ty}^{(2)} \to \mathrm{Ty}$ is (strictly!) a natural transformation.

The isomorphism (14) can now be obtained from λ abstraction for Π, and the fact that Π and π are pointwise isomorphic. $\square$

Definition 2.1.22. A Σ-type structure on $\mathcal{C}$ is given by:

- a term
$$\sigma : \widehat{\mathrm{Tm}}_{\mathrm{Ty}^{(2)}}(\mathrm{Ty}), \tag{15}$$
- an isomorphism
$$\mathrm{Tm}[\sigma] \cong \Sigma_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a] \tag{16}$$
of types over $\mathrm{Ty}^{(2)}$.

Like in the case of Π-type structures, Σ-type structures have a more direct characterisation: giving a Σ-type structure on $\mathcal{C}$ is the same as giving a natural transformation $\sigma : \mathrm{Ty}^{(2)} \to \mathrm{Ty}$, together with a natural isomorphism between $\mathrm{Tm}_\Gamma(\sigma_\Gamma(A, B))$ and the set of pairs (a, b), where $a : \mathrm{Tm}_\Gamma(A)$ and $b : \mathrm{Tm}_\Gamma(B[a])$. Clearly, this is just a reformulation of definition 2.1.12, hence $\mathcal{C}$ supports Σ-types if and only if it has a Σ-type structure.

From this characterisation, we get:

Proposition 2.1.23. *For any category $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is equipped with a canonical Σ-type structure.*

Proof. The morphism $\sigma : \mathrm{Ty}^{(2)} \to \mathrm{Ty}$ can now be taken to be the Σ operation on presheaves, which in this case is automatically natural. The required isomorphism follows directly from the definition of Σ. $\square$

Definition 2.1.24. An equality type structure on $\mathcal{C}$ is given by:

- a term
$$\mathsf{eq} : \widehat{\mathrm{Tm}}_{(A:\mathrm{Ty}).\mathrm{Tm}[A].\mathrm{Tm}[A]}(\mathrm{Ty}), \tag{17}$$
- an isomorphism
$$\mathrm{Tm}[\mathsf{eq}] \cong \mathsf{Eq}_{\mathrm{Tm}[A]} \tag{18}$$
of types over $(A : \mathrm{Ty}).\mathrm{Tm}[A].\mathrm{Tm}[A]$.

By corollary 2.1.17, a term like eq in definition 2.1.24 is given by a map that assigns, to every $A : \mathrm{Ty}(\Gamma)$ a type $\mathsf{eq}(A) : \mathrm{Ty}(\Gamma.A.A)$, naturally in (Γ, A).

Isomorphism (18) is equivalent to an isomorphism between sections of the morphism $\Gamma.A \to \Gamma$ (display map of A), and of the morphism $\Gamma.A.A.\mathsf{eq}(A) \to \Gamma$ (composition of display maps).

Proposition 2.1.25. *For any category $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is equipped with a canonical equality type structure.*

Proof. As for Π and Σ, we want to define eq using the Eq operation on presheaves, but once again we have the problem that Eq, as defined, is not strictly natural. However, thanks to lemma 2.1.10, we can easily define a stricter version of Eq.

For $P : \widehat{\mathcal{C}}$, and $A : \widehat{\mathrm{Ty}}(P)$, let $\mathsf{eq}(A)$ be the image of the unique map $\mathsf{Eq}(A) \to 1$ in $\widehat{\mathbf{Ty}}(P.A.A)$. Since $\mathsf{Eq}(A)$ is subterminal by lemma 2.1.10, it follows that $\mathsf{eq}(A) \cong \mathsf{Eq}(A)$, and eq is clearly natural in A.

The required isomorphism is now easy to construct. □

The construction in proposition 2.1.25 may appear more involved than necessary, since one might be tempted to simply define eq as:

$$\mathsf{eq}(A)_\Gamma(x, a, a') = \begin{cases} 1 & \text{if } a = a' \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

However, a definition like (19) presumes that we are able to decide the equality of arbitrary functions. Classically, (19) is equivalent to the definition given in proposition 2.1.25, but the way we phrased it makes it valid in a constructive setting as well.

Similarly to Π and Σ-type structures, the existence of an equality structure is equivalent to the fact that $\mathcal{C}$ supports equality structures (definition 2.1.13).

Finally, we will define one last structure. This one is fortunately much simpler than the previous three.

Definition 2.1.26. A *unit type structure* on $\mathcal{C}$ is given by:

- a term
$$u : \widehat{\mathrm{Tm}}_1(\mathrm{Ty}) \tag{20}$$
- an isomorphism
$$\mathrm{Tm}[u] \cong 1 \tag{21}$$
of types in the unit context.

And correspondingly:

Proposition 2.1.27. *For any category $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is equipped with a canonical unit type structure.*

Proof. The type u can be set to the unit presheaf 1. The required isomorphism obviously follows from the fact that 1 is terminal. □

Again, unit type structures and the existence of unit types (definition 2.1.14) are equivalent.

2.1.4 *Morphisms*

Given a morphism $F : \mathcal{C} \to \mathcal{D}$ between CwFs, if $\mathcal{C}$ and $\mathcal{D}$ are equipped with one of the structures defined in section 2.1.3, we can ask whether F *preserves* those structures.

Definition 2.1.28. Let $\Gamma : \mathcal{C}$, $(A, B) : \mathrm{Ty}^{(2)}_{\Gamma}$ and $(A', B') : \mathrm{Ty}^{(2)}_{F\Gamma}$. We say that (A, B) and (A', B') are F-related if:

- $FA = A'$
- for all $(\Delta, \sigma) : \mathcal{C}/\Gamma$, and all terms $a : \mathrm{Tm}_{\Delta}(A[\sigma])$, we have that $F(B(a)) = B'(Fa)$.

The following is a direct consequence of definition 2.1.28:

Lemma 2.1.29. *Two pairs (A, B) and (A', B') as in definition 2.1.28 are F-related if and only if:*

- $FA = A'$
- $\phi^F_A(F\widetilde{B}) = \widetilde{B}'$*, where* ϕ^F_A *is as in definition 2.1.4,* $\widetilde{B}$ *is the type in* $\mathrm{Ty}(\Gamma.A)$ *corresponding to B through the isomorphism of corollary 2.1.17, and* $\widetilde{B}'$ *is defined similarly.*

In particular, for all pairs (A, B) in $\mathcal{C}$ there is exactly one pair (A', B') in $\mathcal{D}$ that is related to it. The advantage of formulating the following definitions in terms of related pairs rather than using the characterisation of lemma 2.1.29 directly is that we need no mention of context extension.

Definition 2.1.30. Let (A, B) and (A', B') be F-related pairs, $u : (\Pi_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_{\Gamma}(A, B)$ and $u' : (\Pi_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_{F\Gamma}(A', B')$. We say that u and u' are F-related if for all $(\Delta, \sigma) : \mathcal{C}/\Gamma$, and all terms $a : \mathrm{Tm}_{\Delta}(A[\sigma])$, we have that $F(u(a)) = u'(Fa)$.

Note that the equality between $F(u(a))$ and $u'(Fa)$ in definition 2.1.30 makes sense because (A, B) and (A', B') are themselves related.

Definition 2.1.31. Suppose both $\mathcal{C}$ and $\mathcal{D}$ are equipped with Π-type structures. We say that F *preserves* Π-types if, for all related pairs (A, B) and (A', B'):

- $F(\pi(A, B)) = \pi(A', B')$,
- for all terms $f : \mathrm{Tm}_{\Gamma}(\pi(A, B))$, the element of $(\Pi_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_{\Gamma}(A, B)$ corresponding to f through the Π-type structure on $\mathcal{C}$ is related to the element of $(\Pi_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_{F\Gamma}(A', B')$ corresponding to Ff through the Π-type structure on $\mathcal{D}$.

The definition of preservation of Σ-types is similar, but simpler, because we don't need to define a notion of relatedness for elements of $\Sigma_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a]$, as we can simply map them using F directly:

Definition 2.1.32. Suppose both $\mathcal{C}$ and $\mathcal{D}$ are equipped with Σ-type structures. We say that F *preserves* Σ-types if, for all related pairs (A, B) and (A', B'):

- $F(\sigma(A, B)) = \sigma(A', B')$,
- the following diagram commutes:

$$\begin{tikzcd}
\mathrm{Tm}_\Gamma(\sigma(A,B)) \arrow[r, "\cong"] \arrow[d, "F"'] & (\Sigma_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_\Gamma(A,B) \arrow[d, "F"] \\
\mathrm{Tm}_{F\Gamma}(\sigma(A',B')) \arrow[r, "\cong"] & (\Sigma_{a:\mathrm{Tm}[\mathbf{A}]}\mathrm{Tm}[\mathbf{B}\ a])_{F\Gamma}(A',B'),
\end{tikzcd}$$

  where the horizontal arrows are the isomorphisms given by the Σ-type structures on $\mathcal{C}$ and $\mathcal{D}$ respectively.

For equality types, the definition is entirely analogous:

Definition 2.1.33. Suppose both $\mathcal{C}$ and $\mathcal{D}$ are equipped with equality type structures. We say that F *preserves equality* if, for all $\Gamma : \mathcal{C}$, $A : \mathrm{Ty}(\Gamma)$:

- $F(\mathsf{eq}(A, a, b)) = \mathsf{eq}(FA, Fa, Fb)$,
- the following diagram commutes:

$$\begin{tikzcd}
\mathrm{Tm}_\Gamma(\mathsf{eq}_\Gamma(A,a,b)) \arrow[r, "\cong"] \arrow[d, "F"'] & (\mathsf{Eq}_{\mathrm{Tm}[\mathbf{A}]})_\Gamma(A,a,b) \arrow[d, "F"] \\
\mathrm{Tm}_{F\Gamma}(\mathsf{eq}_{F\Gamma}(FA,Fa,Fb)) \arrow[r, "\cong"'] & (\mathsf{Eq}_{\mathrm{Tm}[\mathbf{A}]})_{F\Gamma}(FA,Fa,Fb)
\end{tikzcd}$$

Finally, we say that F preserves the unit type simply if $Fu = u$ over the unit context.

Replacing equality with isomorphism in the above definitions yields the notions of *weak preservation* of the various type structures.

Remark 2.1.34. Let $F : \mathcal{C} \to \mathcal{D}$ be a CwF morphism. Suppose $\mathcal{C}$ is equipped with a Π structure. Then the application morphism $\epsilon_{A,B} : \Gamma.A.\Pi_A B \to \Gamma.A.B$ can be mapped to $\mathcal{D}$ through F, which implies that we can apply terms of type $F(\Pi_A B)$ to terms of type FA, even though $\mathcal{D}$ might not even have a Π-type structure.

2.1.5 *The Yoneda embedding for CwFs*

If $\mathcal{C}$ is a CwF, the Yoneda embedding $y : \mathcal{C} \to \widehat{\mathcal{C}}$ is a functor between CwFs, so it is natural to ask whether it can be extended to a CwF morphism.

Definition 2.1.35. Let $\Gamma : \mathcal{C}$ be a context, and $A : \mathrm{Ty}(\Gamma)$ a type over Γ. Define the presheaf $y_0(A) : \widehat{\mathrm{Ty}}(y\Gamma)$ as follows:

$$y_0(A)_\Delta(\sigma) :\equiv \mathrm{Tm}_\Delta(A[\sigma]).$$

Proposition 2.1.36. *For all $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, there is a natural isomorphism*

$$y(\Gamma).y_0(A) \cong y(\Gamma.A)$$

over $y(\Gamma)$.

Proof. Immediate consequence of the defining isomorphism of context extension. □

Lemma 2.1.37. *For all $\Gamma : \mathcal{C}$ and $A : \mathrm{Ty}(\Gamma)$, we have:*

$$\mathrm{Tm}_\Gamma(A) \cong \widehat{\mathrm{Tm}}_{y\Gamma}(y_0 A).$$

Proof. It follows from proposition 2.1.36 and proposition 2.1.2 that $\widehat{\mathrm{Tm}}_{y\Gamma}(y_0 A)$ is naturally isomorphic to the set of sections of $y\pi_A : y(\Gamma.A) \to y(\Gamma)$. By the Yoneda lemma, this is isomorphic to the set of sections of $\pi_A : \mathcal{C}(\Gamma.A, \Gamma)$, which, by proposition 2.1.2 again, is isomorphic to $\mathrm{Tm}_\Gamma(A)$. □

Proposition 2.1.38. *For any CwF $\mathcal{C}$, the Yoneda Embedding $y : \mathcal{C} \to \widehat{\mathcal{C}}$ can be extended to a CwF morphism, where y_0 is the action of the morphism on types, and the isomorphism of lemma 2.1.37 is its action on terms.*

Proof. Naturality of y_0 is easy to verify. It only remains to check that the map

$$\phi^y_A : \widehat{\mathcal{C}}(y(\Gamma.A), y(\Gamma).y_0(A))$$

as in definition 2.1.4 is an isomorphism, but this follows immediately from the fact that it is the inverse of the isomorphism of proposition 2.1.36. □

The reason for the subscript 0 in our notation for the action of y on types is that, when $\mathcal{C}$ possesses Π and Σ type structures, the map y_0, as defined, does not preserve them.

We will later define in certain cases a stricter version of y_0 that does indeed preserve the extra structure, and we reserve the name y for that.

2.1.6 *Presheaf universes*

Using a universe of sets Set_i, we can build a universe in any presheaf model. This construction follows closely the one in [19]. Let $\mathcal{C}$ be any small category, and consider the CwF structure on $\widehat{\mathcal{C}}$ defined in section 2.1.2.

Definition 2.1.39. Let P be a context in $\widehat{\mathcal{C}}$. A type $A : \widehat{\mathrm{Ty}}(P)$ is said to be *small* (with respect to Set_i), if it factors through Set_i when regarded as a functor $\int P \to \mathsf{Set}$.

For all object $\Gamma : \mathcal{C}$, let $\mathcal{U}_\Gamma$ be the set of small types over $y\Gamma$. This defines a presheaf $\mathcal{U}$ on $\mathcal{C}$.

For all $\Gamma : \mathcal{C}$ and $P : \mathcal{U}_\Gamma$, define

$$\mathsf{El}_\Gamma(P) :\equiv P_\Gamma(\mathrm{id}).$$

We now have a universe $(\mathcal{U}, \mathsf{El})$ in $\widehat{\mathcal{C}}$.

Proposition 2.1.40. *The universe $(\mathcal{U}, \mathsf{El})$ classifies small types, i.e. a type A over P is small if and only if there exists a term $\widetilde{A}$ of type $\mathcal{U}$ over P such that $A = \mathsf{El}[\widetilde{A}]$.*

Proof. Clearly, El is small, hence $\mathsf{El}[\widetilde{A}]$ is small for all $\widetilde{A} : P \to \mathcal{U}$.

Conversely, if A is small, define $\widetilde{A} : P \to \mathcal{U}$ as follows:

$$\widetilde{A}_\Gamma(x) :\equiv A[x],$$

where $x : y(\Gamma) \to P$ denotes the morphism corresponding to $x : P_\Gamma$ through the isomorphism of the Yoneda lemma. We have:

$$\begin{aligned} \mathsf{El}[\widetilde{A}]_\Gamma(x) &= \mathsf{El}_\Gamma(\widetilde{A}_\Gamma(x)) \\ &= \mathsf{El}_\Gamma(A[x]) \\ &= A[x]_\Gamma(\mathrm{id}) \\ &= A_\Gamma(x). \end{aligned}$$

□

2.1.7 *More notational conventions*

In the following, we will make heavy use of nested Π and Σ types, building complicated type expressions with them. It is therefore convenient to adopt a "flatter" notation, one that is more symmetric with the respect to the two arguments of a Π or Σ type.

This notation is inspired by the syntax of the proof assistant AGDA [28], and it works as follows: a type like $\Pi_{a:A} B$ is written as:

$$(a : A) \to B,$$

mimicking the usual notation for (non-dependent) function types.

Similarly, the type $\Sigma_{a:A}B$ will be written as follows:

$$(a : A) \times B,$$

making it explicit that Σ-types can be thought of as a generalised form of products.

Chained Π types will be written by omitting all the intermediate arrows, and if the same type is present more than once, the corresponding variables can be grouped within one bracket. For example:

$$(a : A)(b, b' : B)(c : C) \to D$$

represents the type:

$$\Pi_{a:A}\Pi_{b:B}\Pi_{b':B}\Pi_{c:C}D.$$

Finally, if $(\mathcal{U}, \mathsf{El})$ is a universe, we will sometimes omit uses of El, as they can be inferred very easily: if a term is used in place of a type, it means that there is an implicit application of El there.

2.1.8 *Fibrations and contextuality*

Definition 2.1.41. Let $p : \mathcal{C}(\Delta, \Gamma)$ be a morphism in a CwF. We say that p is a *fibration* if there is a type $A : \mathrm{Ty}(\Gamma)$ such that p and $p_A : \mathcal{C}(\Gamma.A, \Gamma)$ are isomorphic in the slice category $\mathcal{C}/\Gamma$.

We say that a context Γ is *fibrant* if the unique morphism $\mathcal{C}(\Gamma, 1)$ is a fibration.

Lemma 2.1.42. *In any CwF, pullbacks of fibrations exist and are fibrations.*

Proof. Immediate consequence of proposition 2.1.3. □

Definition 2.1.43. A CwF $\mathcal{C}$ is said to be *contextual* if every context of $\mathcal{C}$ is fibrant.

The idea of definition 2.1.43 is to express the idea that in certain CwFs contexts are none other than types in the unit context. For example, this holds for syntactical models like RF_0, introduced in section 2.2 (see lemma 2.2.6).

If $\mathcal{C}$ is a CwF, and Γ is any context of $\mathcal{C}$, we can put a category structure on $\mathrm{Ty}(\Gamma)$ by defining a morphism between types A and B to be a morphism between p_A and p_B in the slice category $\mathcal{C}/\Gamma$. We denote with $\mathbf{Ty}(\Gamma)$ the resulting category of types over Γ.

Note that the notation $\mathbf{Ty}(\Gamma)$ is consistent with how we denoted the category of types over a presheaf in section 2.1.2.

Proposition 2.1.44. *A CwF $\mathcal{C}$ is contextual if and only if the canonical functor $j : \mathbf{Ty}(1) \to \mathcal{C}$ is an equivalence of categories.*

Proof. The functor j is always fully faithful, and $\mathcal{C}$ being contextual is clearly equivalent to j being essentially surjective. □

Corollary 2.1.45. *A presheaf category is a contextual CwF.*

Contextual CwFs are similar to C-systems (also called contextual categories) [10]. There are, however, two important differences:

- the identification between types and contexts is not canonical, and only up to isomorphism;
- we require that every context can be obtained out of a single type, rather than a chain of types.

In particular, the second condition implies that our notion of contextuality is only well-behaved when $\mathcal{C}$ has a Σ-type structure. It would be possible to formulate definition 2.1.43 in a way that doesn't implicitly require the existence of Σ-types, using the idea of a *telescope* (i.e. a finite sequence of types, each depending on the previous ones), but doing so is cumbersome, and will not be required in the following, so we avoid it.

Proposition 2.1.46. *If $\mathcal{C}$ is a CwF equipped with a Σ-type structure, then the category $\mathbf{Ty}(\Gamma)$ is itself a CwF with a Σ-type structure, and the canonical functor $j : \mathbf{Ty}(\Gamma) \to \mathcal{C}$ is a split CwF morphism preserving Σ-types.*

Proof. Define a type over $A : \mathbf{Ty}(\Gamma)$ to simply be an element of $\mathrm{Ty}(\Gamma.A)$. Context extension and Σ-types can be defined directly using the Σ-type structure of $\mathcal{C}$.

Verifying that j is a split CwF morphism is then straightforward, and the preservation of Σ-types is a direct consequence of the definitions. □

Contextuality has a useful category-theoretic consequence:

Proposition 2.1.47. *Let $\mathcal{C}$ be a contextual CwF. Then $\mathcal{C}$ has finite products.*

Proof. The existence of a terminal object is part of the definition of a CwF, so we only need to show that $\mathcal{C}$ has binary products.

Let $\Gamma, \Delta : \mathcal{C}$ be any two contexts. By contextuality, we can replace Δ with a type A over the unit context. By proposition 2.1.3, the following square is a pullback:

$$\begin{tikzcd}
\Gamma.A \arrow[r] \arrow[d] & A \arrow[d] \\
\Gamma \arrow[r] & 1,
\end{tikzcd}$$

which means that $\Gamma.A$ is the product of Γ and A. □

We conclude this section with a construction that will occasionally be useful later.

Proposition 2.1.48. *Let $\mathcal{C}$ be a CwF, and $\Gamma : \mathcal{C}$ a context. The slice category $\mathcal{C}/\Gamma$ can be equipped with a CwF structure.*

Proof. If (Δ, σ) is an object of $\mathcal{C}$, we simply define types and terms over (Δ, σ) to be the types and terms over Δ in $\mathcal{C}$. □

2.2 THE RULE FRAMEWORK

We will use the type structures defined above to "bootstrap" a more general definition of structure for CwF. To that end, we give the following definition:

Definition 2.2.1. An *RF*-category[2] is a CwF $\mathcal{C}$, equipped with Π, Σ, equality and unit type structures, and a universe $\mathcal{U}$, El. An *RF*-morphism is a CwF morphism preserving all the structure.

RF-categories and *RF*-morphisms form a category $\mathcal{RF}$. Denote by $\mathcal{RF}^{\mathrm{s}}$ the subcategory of $\mathcal{RF}$ consisting of only split morphisms. We will need the following:

Lemma 2.2.2. *The category $\mathcal{RF}^{\mathrm{s}}$ has all small limits.*

Proof. Let I be a small category, and $F : I \to \mathcal{RF}$ a functor. Denote by $\mathcal{C}_i$ the underlying category of $F(i)$.

We construct the limit of F by first taking the limit $\mathcal{C}$ (in Cat) of the $\mathcal{C}_i$, and then defining a CwF structure on $\mathcal{C}$, equipped with all the required type structures.

For a context $\Gamma : \mathcal{C}$, denote by Γ_i the context of $\mathcal{C}_i$ obtained from Γ through the projection of the universal cone $\mathcal{C} \to \mathcal{C}_i$. Types over Γ are defined to be simply the limit of $\mathrm{Ty}(\Gamma_i)$ over I.

Similarly, if A is a type over Γ, we write A_i for the projection of A to $\mathrm{Ty}(\Gamma_i)$, and define terms of type A as the limit of $\mathrm{Tm}_{\Gamma_i}(A_i)$.

Context extension is defined pointwise. This is the crucial point where we use the fact that the diagram F is composed solely of split morphisms.

Verifying that this gives a CwF structure on $\mathcal{C}$ is straightforward.

As for the Π, Σ, equality and unit type structures, they can all be defined pointwise, and the resulting RF-category is easily seen to satisfy the universal property of the limit. □

Theorem 2.2.3. *The category $\mathcal{RF}^{\mathrm{s}}$ has an initial object RF_0.*

2 *RF* stands for *rule framework*

Theorem 2.2.3 can be proved by giving an explicit inductive definition of RF_0: types are expressions generated from base types like $\mathcal{U}$, El and the unit type, by applying the operations of the RF structures: Π, Σ and equality. Similarly, terms are generated from variables and their weakening by applying the various isomorphisms of the RF structures. Contexts are defined as tuples of types, and morphisms as tuples of terms.

Making this sort of definition precise is, however, far from a straightforward task, as is proving that it in fact gives an initial object of $\mathcal{RF}$. Intuitively, initiality follows because we can regard every context (resp. type, term, morphism) in RF_0 as a "recipe" to build a context (resp. type, term, morphism) in an arbitrary RF-category $\mathcal{C}$. This gives, for any such $\mathcal{C}$, a uniquely determined functor $RF_0 \to \mathcal{C}$ that clearly preserves all the structures.

We follow a slightly more indirect approach, based on the ideas underlying the proof of the adjoint functor theorem. Indeed, the following proof could be adapted to show the more general fact that the forgetful functor $\mathcal{RF} \to \mathsf{Cat}$ has a left adjoint. However, we will not need the extra generality.

Proof of theorem 2.2.3. Since $\mathcal{RF}^{\mathrm{s}}$ has all small limits (lemma 2.2.2), it is enough to show that it has a weakly-initial family. We say that a small RF-category is *countable* if the set of objects is countable, all the homsets are countable, and $\mathrm{Ty}(\Gamma)$ and $\mathrm{Tm}_\Gamma(A)$ are countable for all Γ and A.

We will show that every RF-category contains a countable RF-subcategory. From this fact, the existence of a weakly-initial family easily follows (for example, fix a countably infinite set Ω and take the family of all RF-categories whose contexts, morphisms, types and terms are all elements of Ω).

Let $\mathcal{C}$ be an RF-category. We define a chain of subsets $\mathcal{D}_n$ of $\mathcal{C}$, each equipped with subfamilies of morphisms, types and terms, arranged just like in a CwF, but with no further structure. The morphisms of $\mathcal{D}_n$ between contexts Δ and Γ will be denoted $\mathcal{D}_n(\Delta, \Gamma)$, just like in a category, and they will form a subset of $\mathcal{C}(\Delta, \Gamma)$. We will write $\mathrm{Ty}^n(\Gamma)$ for the types of $\mathcal{D}_n$ over Γ, which will form a subset of $\mathrm{Ty}(\Gamma)$, and similarly for terms.

The starting point $\mathcal{D}_0$ is just the empty subset. Given $\mathcal{D}_n$ and its associated structures, define $\mathcal{D}_{n+1}$ as the subset of $\mathcal{C}$ containing $\mathcal{D}_n$, plus all the contexts, morphisms, types and terms that are obtained from those of $\mathcal{D}_n$ by applying any of the operations of the RF-category $\mathcal{C}$. In detail:

- the set $\mathcal{D}_{n+1}$ contains all the elements of $\mathcal{D}_n$, plus the unit context, and the context $\Gamma.A$, for all choices of $\Gamma : \mathcal{D}_n$ and $A : \mathrm{Ty}^n(\Gamma)$;
- morphisms of $\mathcal{D}_{n+1}$ are obtained from those of $\mathcal{D}_n$ by adding the canonical morphism to the unit context, identity morphisms, compositions of morphisms in $\mathcal{D}_n$, projections of types in $\mathcal{D}_n$ and substitutions of the form $\langle \sigma, a \rangle$, where $\sigma : \mathcal{D}_n(\Delta, \Gamma)$, and $a : \mathrm{Tm}^n_\Delta(A[\sigma])$;

- the set $\mathrm{Ty}^{n+1}(\Gamma)$ contains all the types of $\mathcal{D}_n$, plus the unit type, types of the form $\Pi_A B$ and $\Sigma_A B$, where $A : \mathrm{Ty}^n(\Gamma)$, and types of the form $a = b$, where $a, b : \mathrm{Tm}^n_\Gamma(A)$, and $A : \mathrm{Ty}^n(\Gamma)$;
- the set $\mathrm{Tm}^{n+1}(\Gamma)$ contains all the terms of $\mathcal{D}_n$, plus the unique inhabitant of the unit type, and the images of the isomoprhisms defining Π, Σ and equality types and their inverses.

From the fact that every operation in the definition of *RF*-category has a finite number of arguments, it easily follows that the union of all the $\mathcal{D}_n$ and corresponding structures forms an *RF*-subcategory of $\mathcal{C}$. □

The advantage of the proof above over the usual technique of building the initial model purely syntactically is that the iterative construction happens within an existing CwF, hence we only need to concern ourselves with adding the necessary elements to the structures involved, and their required properties will automatically hold, because they do so in the ambient category.

We will write RF_0 to denote the initial object of $\mathcal{RF}^{\mathrm{s}}$. Since RF_0 is only initial in a subcategory of $\mathcal{RF}$, we cannot conclude that it is initial in $\mathcal{RF}$. In particular, given an *RF*-category $\mathcal{C}$, we can always give a morphism $RF_0 \to \mathcal{C}$, but that morphism might not be unique.

Fortunately, we can prove a weaker version of uniqueness.

Definition 2.2.4. A *weak RF-morphism* is a CwF morphism that weakly preserves all the structure.

Theorem 2.2.5. *Let $\mathcal{C}$ be an RF-category, and $F, G : RF_0 \to \mathcal{C}$ two weak RF-morphisms in $\mathcal{RF}$. Then F and G are isomorphic.*

Proof. Construct an *RF*-category $\mathcal{E}$ (the *pseudo-equaliser* of F and G) as follows: the objects of $\mathcal{E}$ are contexts Γ in RF_0, together with an isomorphism between $F\Gamma$ and $G\Gamma$. Similarly, types (resp. terms) in $\mathcal{E}$ are types (resp. terms) in RF_0, together with an isomorphism between their respective images in $\mathcal{C}$.

The fact that F and G are weak *RF*-morphisms implies that it is possible to equip $\mathcal{E}$ with a structure of *RF*-category such that the obvious projection $\pi : \mathcal{E} \to RF_0$ is a split morphism.

By initiality of $\mathcal{E}$, the morphism π has a section, which implies that F and G are isomorphic. □

Lemma 2.2.6. *The category RF_0 is contextual.*

Proof. It is easy to see that $\mathbf{Ty}(1)$ can be equipped with an *RF*-category structure such that the canonical functor $j : \mathbf{Ty}(1) \to RF_0$ is a split *RF*-morphism (see proposition 2.1.46). It follows that j is an isomorphism of *RF*-categories, hence RF_0 is contextual by proposition 2.1.44. □

2.3 TYPE FORMERS AND STRUCTURES

We know from section 2.1 that presheaf categories are equipped with a canonical CwF structure, as well as Π, Σ, equality and unit type structures. If $\mathcal{C}$ is a CwF, then its presheaf category additionally possesses a canonical universe (the *fibrant* universe) given by the presheaves of types and terms. Therefore, we have that for any CwF $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is an RF-category.

Definition 2.3.1. A *type former* is a context in RF_0.

The idea behind definition 2.3.1 is that we can use the language of RF_0 as a meta-theoretical framework to describe structures on a generic CwF $\mathcal{C}$. The universe $\mathcal{U}$ in RF_0 intuitively stands for the collection of types of $\mathcal{C}$. Given some $A : \mathcal{U}$, the RF-type $\mathsf{El}[A]$ corresponds to the terms of A regarded as a type on $\mathcal{C}$.

Making this intuition precise is relatively straightforward: denote by $[\![-]\!]^{\widehat{\mathcal{C}}}$ the unique split morphism $RF_0 \to \widehat{\mathcal{C}}$. Using $[\![-]\!]^{\widehat{\mathcal{C}}}$, any type former can be interpreted as a presheaf on $\mathcal{C}$ constructed from Ty and Tm, using the operations of RF-categories in $\widehat{\mathcal{C}}$.

Definition 2.3.2. Let Φ be a type former, and $\mathcal{C}$ a CwF. A Φ-structure on on $\mathcal{C}$ is a global element of $[\![\Phi]\!]^{\widehat{\mathcal{C}}}$. A CwF equipped with a Φ-structure will be referred to as a Φ-CwF.

Lemma 2.3.3. *Let Φ be a type former. A Φ-structure ϕ on $\mathcal{C}$ can be transported to a Φ-structure on the slice category $\mathcal{C}/\Gamma$ for any context Γ.*

Proof. The Φ-structure ϕ can be regarded as a term of type $[\![\Phi]\!]$ in the unit context of $\widehat{\mathcal{C}}$. If $! : y(\Gamma) \to 1$ is the unique morphism to the terminal object of $\widehat{\mathcal{C}}$, it is not hard to verify that $[\![\Phi]\!][!]$ coincides with $[\![\Phi]\!]^{\widehat{\mathcal{C}/\Gamma}}$ under the isomorphism of proposition 2.1.7. Therefore, $\phi[!]$ is a Φ-structure for $\mathcal{C}/\Gamma$. $\square$

It follows from lemma 2.3.3 that a slice of an RF-category is itself an RF-category.

2.4 EXAMPLES

All of the commonly employed type structures on CwFs can be expressed using the notion of type former developed in section 2.3.

In particular, we can now revisit the definitions of the type structures of an RF-category, as given in section 2.1, and reformulate them in terms of type formers.

For example, a Π-type structure is none other than a Φ^{Π}-structure, where Π is the following type former:

$$\begin{aligned}\Phi^{\Pi} = {} & (A : \mathcal{U})(B : A \to \mathcal{U}) \\ & \to (P : \mathcal{U}) \times (P \cong ((a : A) \to B\ a)),\end{aligned} \tag{22}$$

where we are making use of the notation described in section 2.1.7 to represent nested Π and Σ types in RF, and uses of El are implicit. The symbol $\cong$ refers to a notion of *isomorphism* internal to RF, defined in the natural way:

$$\begin{aligned}A \cong B :\equiv {} & (f : A \to B) \\ & \times (g : B \to A) \\ & \times ((x : A) \to g(f(x)) = x) \\ & \times ((y : B) \to f(g(y)) = y).\end{aligned}$$

Note that the equality symbol used here and in following type formers refers to the equality type structure that is part of the definition of RF-category.

Expanding the definition of isomorphism into (22) brings it closer to the traditional formulation of Π-types: the return Σ-type in (22) consists of five components, corresponding to the formation, elimination and introduction rule, plus β and η equalities [18].

Similarly, we can define a type former Φ^{Σ} for Σ-type structures, a type former Φ^{EQ} for equality type structures, and a type former Φ^{UNIT} for unit type structures.

Unfortunately, we cannot use the above characterisations as definitions, because we need to bootstrap the process with a number of basic type structures in order to define RF_0.

However, we can now use RF to give succinct definitions of other commonly employed type structures, and, more importantly, we can prove metatheoretical results on CwFs while remaining agnostic of the particular type structures that they carry.

One of the simplest examples that we haven't covered directly so far is given by *binary sums*. They can be defined by the following *RF* type former:

$$\begin{aligned}\Phi^{\text{SUM}} :\equiv\ & (A, B : \mathcal{U})\\ & \to (S : \mathcal{U})\\ & \times (l : A \to S)\\ & \times (r : B \to S)\\ & \times ((P : S \to \mathcal{U})\\ & \quad (d_1 : (x : A) \to P(l(x)))\\ & \quad (d_2 : (y : B) \to P(r(y)))\\ & \quad (f : (s : S) \to P(s))\\ & \quad \times ((x : A) \to f(l(x)) = d_1(x))\\ & \quad \times ((y : B) \to f(r(y)) = d_2(y))).\end{aligned}$$

Another important example is *intensional equality*, the cornerstone of Martin-Löf type theory, and HoTT in particular. This is not to be confused with the equality type former introduced in section 2.1.3.

$$\begin{aligned}\Phi^{\text{IEQ}} :\equiv\ & (A : \mathcal{U})\\ & \to (E : A \to A \to \mathcal{U})\\ & \times (r : (a : A) \to E(a, a))\\ & \times (((a : A)(P : (b : A) \to E(a, b) \to \mathcal{U})\\ & \quad (d : P(a, r(a)))\\ & \quad \to (J : (b : A)(p : E(a, b)) \to P(b, p))\\ & \quad \times J(a, r(a)) = d).\end{aligned} \tag{23}$$

For comparison, the *extensional* equality type structure of section 2.1.3 can be represented in *RF* as follows:

$$\begin{aligned}\Phi^{\text{EQ}} :\equiv\ & (A : \mathcal{U})\\ & \to (E : A \to A \to \mathcal{U})\\ & \times ((a, b : A) \to E(a, b) \cong (a = b)).\end{aligned}$$

Given a Φ^{EQ}-structure, the ability to convert any *propositional* equality, (i.e. a term of type $E(a, b)$), into a *definitional* equality (i.e. an equality of a and b as terms), is often referred to as the "reflection rule".

The difference between intensional and extensional equality can then be summarised by the statement that intensional equality does not admit a reflection rule, and instead replaces it with the J *eliminator* and corresponding computation rule given in (23).

We employed extensional equality as a very convenient technical device in the development of our framework of type formers. Indeed, many "natural" models of type theory like Set or any presheaf model come equipped with a straightforward extensional equality structure.

However, in a constructive setting, extensional equality has certain undesirable characteristics (for example, models with extensional equality, such as RF_0, tend to have undecidable equality of terms), hence intensional equality is often preferred.

As a compromise between the two forms of equality, we recall the following rule, depending on some $E : \Phi^{\mathrm{IEQ}}$, called *uniqueness of identity proofs* (UIP).

$$\begin{aligned} \mathrm{UIP}(E) :\equiv\ & (A : \mathcal{U}) \\ & \to (a, b : A) \\ & \to (p, q : E(a, b)) \\ & \to E(p, q). \end{aligned} \tag{24}$$

UIP says that any two parallel equalities are themselves equal, which means that types do not possess any higher equality structure. In HoTT terminology, this can be expressed by saying that *every type is a set.*

We will refer to the type former $(E : \Phi^{\mathrm{IEQ}}) \times \Phi^{\mathrm{UIP}}(E)$ as *strict equality.* Note that extensional equality satisfies UIP, hence it can be regarded as a special case of strict equality.

Other type formers that we will need in the following are:

- Φ^{EMPTY}, for the empty type;
- $\Phi^{\mathbb{N}}$, for the natural numbers;
- Φ^{FUNEXT}, for *function extensionality.*

Their definitions can be obtained by encoding in RF_0 the usual rules that concern them. See for example [36] for a detailed exposition of these type formers and similar ones.

2.5 MORPHISMS

Similarly to what we did in section 2.1.4, we want to define what it means for a morphism between CwFs $F : \mathcal{C} \to \mathcal{D}$ to *preserve* a Φ-structure. Unfortunately, due to the presence of Π-types in the description of a type former as a context in RF_0, this turns out to be quite challenging.

In fact, given a CwF morphism $F : \mathcal{C} \to \mathcal{D}$, it is not possible in general to define a corresponding RF-morphism $\widetilde{F}$ between $\widehat{\mathcal{C}}$ and $\widehat{\mathcal{D}}$, in either direction. If we had such a morphism, we could say that F *preserves* Φ-structures when $\widetilde{F}$ maps the Φ-structure on $\mathcal{C}$ into the one on $\mathcal{D}$, or vice versa.

However, since this is not the case, our definition of preservation of type structures is much more cumbersome, and requires setting up some infrastructure to be able to talk about a form of "logical relations" on type structures. Then, given an F, we will be able to recursively define the preservation relation on Φ-structures on $\mathcal{C}$ and $\mathcal{D}$, essentially by induction on Φ.

Definition 2.5.1. An *oplax RF-morphism* between RF-categories $\mathcal{A}$ and $\mathcal{A}'$, with universes $(\mathcal{U}, \mathsf{El})$ and $(\mathcal{U}', \mathsf{El}')$ respectively, is given by:

- a CwF morphism $F : \mathcal{A} \to \mathcal{A}'$;
- a morphism $F^{\mathcal{U}} : \mathcal{A}'(\mathcal{U}', F\mathcal{U})$;
- a morphism $F^{\mathsf{El}} : \mathcal{A}'(\mathcal{U}'.\mathsf{El}', F(\mathcal{U}.\mathsf{El}))$;

such that the following diagram commutes:

$$\begin{tikzcd}
\mathcal{U}'.\mathsf{El}' \arrow[r, "F^{\mathsf{El}}"] \arrow[d] & F(\mathcal{U}.\mathsf{El}) \arrow[d] \\
\mathcal{U}' \arrow[r, "F^{\mathcal{U}}"'] & F\mathcal{U}.
\end{tikzcd}$$

We will often suppress the superscript $\mathcal{U}$ and El from our notation when working with an oplax RF-morphism.

Note that F is *not* required to preserve any of the RF-structure.

Given an oplax RF-morphism $F : \mathcal{A} \to \mathcal{A}'$, we can construct an RF-category $\mathcal{R}_F$.

Objects of $\mathcal{R}_F$ are defined to be triples $\mathbf{\Gamma} = (\Gamma, \Gamma', R)$, where $\Gamma : \mathcal{A}$, $\Gamma' : \mathcal{A}'$, and R is a *span* over $F\Gamma$ and Γ', i.e. a diagram in $\mathcal{A}'$ of the form:

$$\begin{tikzcd}
F\Gamma & R \arrow[l, "l"'] \arrow[r, "r"] & \Gamma'.
\end{tikzcd} \tag{25}$$

A type over $\mathbf{\Gamma}$ is itself a triple $\mathbf{A} = (A, A', X)$, where $A : \mathrm{Ty}(\Gamma)$, $A' : \mathrm{Ty}(\Gamma')$, and $X : \mathrm{Ty}(R.FA[l].A'[r])$. Context extension of (A', A', X) is defined to be the span determined by $R.FA[l].A'[r].X$.

Terms of type $\mathbf{A}$ are defined to be triples (a, a', x), where $a : \mathrm{Tm}_\Gamma(A)$, $a' : \mathrm{Tm}_{\Gamma'}(A')$, and $x : \mathrm{Tm}_R(X[al, a'r])$.

This determines a CwF structure on $\mathcal{R}_F$, and it is easy to see that the two obvious projections $\mathcal{R}_F \to \mathcal{A}$ and $\mathcal{R}_F \to \mathcal{A}'$ are split CwF morphisms.

Proposition 2.5.2. *The CwF $\mathcal{R}_F$ defined above has an RF-structure, and the two projections $\mathcal{R}_F \to \mathcal{A}$ and $\mathcal{R}_F \to \mathcal{A}'$ are split RF-morphisms.*

Proof. We will only show how to define a Π-type structure on $\mathcal{R}_F$, since this is the most involved step.

Let $\mathbf{\Gamma} = (\Gamma, \Gamma', R)$ be a context in $\mathcal{R}_F$, $\mathbf{A} = (A, A', X)$ a type over it, and $\mathbf{B} = (B, B', Y)$ a type over $\mathbf{\Gamma}.\mathbf{A}$. Let R be given by the span in (25).

The Π-type $\Pi_{\mathbf{A}}\mathbf{B}$ is defined as the triple $\mathbf{P} = (\Pi_A B, \Pi_{A'} B', P)$, where P is the following type in the context $R_0 = R(u : F(\Pi_A B)[l])(u' : \Pi_{A'} B'[r])$:

$$\Pi_{a:FA[l]}\Pi_{a':A'[r]}\Pi_{X[a,a']}Y[a, u\ a, a', u'\ a'],$$

and $u\ a$ denotes the application of $u : F(\Pi_A B)[l]$ to $a : FA[l]$, as described in remark 2.1.34.

Terms of type $\mathbf{P}$ are triples (u, u', w), where $u : \mathrm{Tm}_\Gamma(\Pi_A B)$, $u' : \mathrm{Tm}_\Gamma(\Pi_{A'} B')$, and $w : \mathrm{Tm}_R(P[Fu[l], u'r])$.

Using the defining properties of Π-type structures in $\mathcal{A}$ and $\mathcal{A}'$, we can see that these are naturally isomorphism to triples (b, b', y), where $b : \mathrm{Tm}_{\Gamma.A}B$, $b' : \mathrm{Tm}_{\Gamma'.A'}B'$, and $y : \mathrm{Tm}_{R(a:FA[l])(a':A'[r]).X}(Y[a, Fb[l], a', b'[r]])$, which are exactly terms of type $\mathbf{B}$ in the context $\mathbf{\Gamma}.\mathbf{A}$. □

Since the functors $\mathcal{R}_F \to \mathcal{A}$ and $\mathcal{R}_F \to \mathcal{A}'$ are split RF-morphisms by proposition 2.5.2, initiality of RF_0 implies that $[\![\Phi]\!]^{\mathcal{R}_F}$ is a span over $[\![\Phi]\!]^{\mathcal{A}}$ and $[\![\Phi]\!]^{\mathcal{A}'}$. We will write that span as:

$$\begin{tikzcd} F[\![\Phi]\!]^{\mathcal{A}} & {[\![\Phi]\!]^{F}} \arrow[l] \arrow[r] & {[\![\Phi]\!]^{\mathcal{A}'}}. \end{tikzcd}$$

Definition 2.5.3. Let ϕ be a global element of $[\![\Phi]\!]^{\mathcal{A}}$ and ϕ' a global element of $[\![\Phi]\!]^{\mathcal{A}'}$. An *element* of $[\![\Phi]\!]^{F}$ over ϕ and ϕ' is defined to be a global element s of $[\![\Phi]\!]^{F}$ such that the following diagram commutes:

$$\begin{tikzcd} & 1 \arrow[dl, "\phi"'] \arrow[d, "s"] \arrow[dr, "\phi'"] & \\ F[\![\Phi]\!]^{\mathcal{A}} & {[\![\Phi]\!]^{F}} \arrow[l] \arrow[r] & {[\![\Phi]\!]^{\mathcal{A}'}}. \end{tikzcd}$$

Let us now fix two CwFs $\mathcal{C}$ and $\mathcal{D}$, both equipped with $\mathbf{\Phi}$-structures for some type former $\mathbf{\Phi}$, and a CwF morphism $F : \mathcal{C} \to \mathcal{D}$. The following lemma is an immediate consequence of definition 2.5.1.

Lemma 2.5.4. *The functor $F^* : \widehat{\mathcal{D}} \to \widehat{\mathcal{C}}$ is an oplax RF-morphism.*

It follows from lemma 2.5.4 that we have an RF-category $\mathcal{R}_{F^*}$, thus we get an interpretation morphism $RF_0 \to \mathcal{R}_{F^*}$.

Definition 2.5.5. Let ϕ and ψ be the $\mathbf{\Phi}$-structures of $\mathcal{C}$ and $\mathcal{D}$ respectively. We say that F is a $\mathbf{\Phi}$-*morphism* (or that F preserves $\mathbf{\Phi}$-structures) if there exists an element of $[\![\Phi]\!]^{F^*}$ over ϕ and ψ.

Definition 2.5.5 is based on the idea of *logical relations* [35]. For a fixed CwF morphism F, we defined a notion of "being related through F" for $\mathbf{\Phi}$-structures, by induction on $\mathbf{\Phi}$.

For the type formers of RF itself, it is not hard to see that preservation as defined in section 2.1.4 coincides with the notion of definition 2.5.5, when using the equivalent definitions given in section 2.4.

Note that, for a general $\mathbf{\Phi}$, for $\mathbf{\Phi}$-structures ϕ and ψ on $\mathcal{C}$ and $\mathcal{D}$ respectively, being related through F does not mean that ϕ can be mapped through F to a $\mathbf{\Phi}$-structure on $\mathcal{D}$ that happens to coincide with ψ. In fact, there is no way in general to transport a $\mathbf{\Phi}$-structure along an arbitrary functor.

This can be understood in analogy with common algebraic structures. For example, given two monoids A and B, and a function between them $f : A \to B$, we know what it means for f to be a monoid homomorphism - meaning that the two monoid structures on A and B are "related through f" - but there is in general no way to transport a monoid structure from A to B.

2.6 COMPOSITION OF MORPHISMS

Unfortunately, for a general type former $\mathbf{\Phi}$, definition 2.5.5 is not very well behaved. In fact, it is not even guaranteed that composition of $\mathbf{\Phi}$-morphisms is a $\mathbf{\Phi}$-morphism, that is, $\mathbf{\Phi}$-CwFs do not necessarily form a category.

The problem becomes apparent as soon as we consider certain "higher order" type formers, i.e. type formers with Π types nested on the left. The simplest example is:

$$\mathbf{\Phi} :\equiv (\mathcal{U} \to \mathcal{U}) \to \mathcal{U}.$$

To make our example easier to follow, we observe that, given any set A, we can construct a CwF with only one context 1, $\mathrm{Ty}(1) = A$, and $\mathrm{Tm}_1(a) = 1$ for all types $a : A$, with context extension defined in the only possible way.

If we assume that the set A is equipped with a function $A^A \to A$, then its corresponding CwF can be equipped with a $\mathbf{\Phi}$-structure. Let us call a set equipped with such a structure a $\mathbf{\Phi}$-set.

Given a function $f : A \to B$ between $\mathbf{\Phi}$-sets, we say that it is a $\mathbf{\Phi}$-morphism if it induces a $\mathbf{\Phi}$-morphism on the corresponding $\mathbf{\Phi}$-CwFs. If we denote by ϕ_A and ϕ_B the $\mathbf{\Phi}$-structures on A and B respectively, what this means is that for all functions $u : A \to A$ and $v : B \to B$ such that the following diagram commutes:

$$\begin{tikzcd} A \arrow[r, "u"] \arrow[d, "f"'] & A \arrow[d, "f"] \\ B \arrow[r, "v"'] & B, \end{tikzcd}$$

we have that $f(\phi_A(u)) = \phi_B(v)$.

To show that $\mathbf{\Phi}$-morphisms between $\mathbf{\Phi}$-CwFs are not in general closed under composition, it is therefore enough to find $\mathbf{\Phi}$-morphisms $f : A \to B$, $g : B \to C$ such that $g \circ f$ is not a $\mathbf{\Phi}$-morphism.

We take $A = 1$, $B = 2$, $C = 3$, and f, g to be inclusions. The $\mathbf{\Phi}$ structure ϕ_A on A is the only possible one, while the $\mathbf{\Phi}$-structure ϕ_B on B takes a function $u : B \to B$ and returns $u(0)$.

The $\mathbf{\Phi}$-structure ϕ_C on C is defined as follows: given $u : C \to C$, it distinguishes two cases:

- if $u(2) \subseteq 2$, then $\phi_C(u) = u(0)$;
- otherwise, $\phi_C(u) = 1$.

It is easy to see that the inclusions $A \to B$ and $B \to C$ are indeed $\mathbf{\Phi}$-morphisms. However, if we take for example the function $u : C \to C$ that swaps 1 and 2 and fixes 0, then clearly the following diagram commutes:

$$\begin{tikzcd}
1 \arrow[r, "\mathrm{id}"] \arrow[d, "0"'] & 1 \arrow[d, "0"] \\
C \arrow[r, "u"'] & C,
\end{tikzcd}$$

but $\phi_C(v) = 1 \neq 0 = \phi_A(\mathrm{id})$.

2.7 SPECIAL TYPE FORMERS

The notion of type formers is very general. As shown in section 2.6, it is possible to define "higher order" type formers, for which even the most basic properties are not provable.

In practice, most of the commonly employed type formers are much better behaved than in the general case. For this reason, it is useful to single out certain specific properties of type formers that make them more suitable to be analysed.

Lemma 2.7.1. *Let $\mathbf{\Phi}$ be a type former, $F : \mathcal{A} \to \mathcal{A}'$ an oplax RF-morphism. Suppose ϕ is a global element of $[\![\mathbf{\Phi}]\!]^{\mathcal{A}}$ and ϕ' a global element of $[\![\mathbf{\Phi}]\!]^{\mathcal{A}'}$. Then any two elements of $[\![\mathbf{\Phi}]\!]^F$ over ϕ and ϕ' are equal.*

Proof. Let $\mathcal{R}'_F$ be subcategory of $\mathcal{R}_F$ consisting of all those objects

$$\begin{tikzcd}
F\Gamma & R \arrow[l, "l"'] \arrow[r, "r"] & \Gamma',
\end{tikzcd}$$

where R is subterminal in the category of spans over $F\Gamma$ and Γ'.

It is not hard to see that $\mathcal{R}'_F$ is itself an RF-category, and consequently the inclusion functor $i : \mathcal{R}'_F \to \mathcal{R}_F$ is a split RF-morphism.

It follows that the interpretation functor $[\![-]\!]^{\mathcal{R}_F}$ has values in $\mathcal{R}'_F$. In particular, $[\![\Phi]\!]^F$ is subterminal over $F[\![\Phi]\!]^{\mathcal{A}}$ and $[\![\Phi]\!]^{\mathcal{A}}$, which is exactly what we had to prove. □

Lemma 2.7.1 ensures that, if a CwF morphism $F : \mathcal{C} \to \mathcal{D}$ between Φ-CwFs is a Φ-morphism, then there is at most one possible choice for the element s of definition 2.5.5.

Now, given oplax RF-morphisms $F : \mathcal{A} \to \mathcal{B}$ and $G : \mathcal{B} \to \mathcal{C}$, we can form the pullback $\mathcal{R}_F \times_{\mathcal{B}} \mathcal{R}_G$, which is an RF-category by lemma 2.2.2, and is equipped with split morphisms π_A and π_C to $\mathcal{A}$ and $\mathcal{C}$ respectively.

Definition 2.7.2. We say that a type former Φ is *flat* if for all F, G as above, whenever $[\![\Phi]\!]^{\mathcal{R}_F \times_{\mathcal{B}} \mathcal{R}_G}$ has a global element s, then there is an element of $[\![\Phi]\!]^{GF}$ over $\pi_A(s)$ and $\pi_C(s)$.

Definition 2.7.2 formalises the idea of a type former that is well-behaved with respect to composition, as the following proposition shows.

Proposition 2.7.3. *If Φ is a flat type former, composition of Φ-morphisms is a Φ-morphism.*

Proof. If $F : \mathcal{A} \to \mathcal{B}$ and $G : \mathcal{B} \to \mathcal{C}$ are Φ-morphisms, then $[\![\Phi]\!]^{\mathcal{R}_F \times_{\mathcal{B}} \mathcal{R}_G}$ has a global element s, where $\pi_A(s)$ is the Φ-structure on $\mathcal{A}$, and $\pi_B(s)$ is the Φ-structure on $\mathcal{C}$.

Since Φ is flat, we get a corresponding element of $[\![\Phi]\!]^{GF}$, showing that GF is a Φ-morphism. □

Corollary 2.7.4. *Let Φ be a flat type former. Φ-CwFs, together with Φ-morphisms, form a category.*

Definition 2.7.5. A flat type former Φ is said to be *algebraic* if the category of Φ-CwFs and *split* Φ-morphism has an initial object.

All the usually considered type formers are algebraic. In particular, all the type formers involved in the definition of an RF-category are algebraic (as essentially proved by theorem 2.2.3), as well as all the examples of section 2.4.

Proposition 2.7.6. *Let Φ be an algebraic type former, $\mathcal{H}$ the initial Φ-CwF, and $\mathcal{C}$ an arbitrary Φ-CwF. Then any two Φ-morphisms $F, G : \mathcal{H} \to \mathcal{C}$ are isomorphic.*

Proof. Let $\mathcal{S}$ be the RF-category whose objects are triples (P, Q, R), where P is a presheaf on $\mathcal{H}$, Q a presheaf on $\mathcal{C}$, and R a span of the form:

$$\begin{tikzcd} F^*Q & R \arrow[l] \arrow[r] & G^*Q. \end{tikzcd}$$

Let $\mathcal{E}$ the pseudo-equaliser of F and G, defined like in the proof of theorem 2.2.5. Let $\pi : \mathcal{E} \to \mathcal{H}$ be the canonical projection.

We can define RF-morphisms

$$\begin{tikzcd} \mathcal{R}_{F^*} \times_{\widehat{\mathcal{C}}} \mathcal{R}_{G^*} \arrow[r] & \mathcal{S} \arrow[r] & \mathcal{R}_{\pi^*}. \end{tikzcd}$$

The fact that F and G are both $\mathbf{\Phi}$-morphisms determines a global element of the interpretation of $\mathbf{\Phi}$ in $\mathcal{R}_{F^*} \times_{\widehat{\mathcal{C}}} \mathcal{R}_{G^*}$, which can therefore be transported to $\mathcal{R}_{\pi^*}$.

It follows that $\mathcal{E}$ can be equipped with a $\mathbf{\Phi}$-structure such that the CwF morphism π is a split $\mathbf{\Phi}$-morphism. The conclusion now follows immediately from the initiality of $\mathcal{H}$. □

Definition 2.7.7. A type former $\mathbf{\Phi}$ is said to be *set-theoretic* if for all small categories $\mathcal{A}$, the CwF $\widehat{\mathcal{A}}$ has a $\mathbf{\Phi}$-structure.

Again, all type formers considered so far are set-theoretic. In section 2.8 we will define a type former for a *univalent universe* (definition 2.8.6), which fails to be set-theoretic.

A type in RF_0 over some type former $\mathbf{\Psi}$ will be referred to as a *type former over* $\mathbf{\Psi}$. Given such a type $\mathbf{\Phi}$, we will often identify it with the corresponding context extension $\mathbf{\Psi}.\mathbf{\Phi}$.

Definition 2.7.8. Let $\mathbf{\Psi}$ be a type former, $\mathbf{\Phi}$ a type former over $\mathbf{\Psi}$, and $\mathcal{C}$ a CwF equipped with a $\mathbf{\Psi}$-structure ψ.

A $\mathbf{\Phi}$*-structure* on $\mathcal{C}$ is a $\mathbf{\Psi}.\mathbf{\Phi}$-structure on the underlying CwF such that the induced $\mathbf{\Psi}$ structure is equal to ψ.

2.8 SYSTEMS OF UNIVERSES

If $(\mathcal{U}, \mathsf{El})$ is a universe in a CwF $\mathcal{C}$, $\mathcal{U}$ induces another CwF structure on $\mathcal{C}$, which we shall denote with the superscript $\mathcal{U}$. Types of $\mathcal{C}^{\mathcal{U}}$ over a context Γ are defined by:

$$\mathrm{Ty}^{\mathcal{U}}(\Gamma) :\equiv \mathcal{C}(\Gamma, \mathcal{U}).$$

For a type $A : \mathrm{Ty}^{\mathcal{U}}(\Gamma)$, we define terms of A as follows:

$$\mathrm{Tm}^{\mathcal{U}}_{\Gamma}(A) :\equiv \mathcal{C}(\Gamma, \mathsf{El}[A]).$$

There is a canonical map $\mathcal{C}^{\mathcal{U}} \to \mathcal{C}$, which is easily verified to be a CwF morphism.

Definition 2.8.1. Let $(\mathcal{U}, \mathsf{El})$ and $(\mathcal{U}', \mathsf{El}')$ be universes in a CwF $\mathcal{C}$. A *universe morphism* $\mathcal{U} \to \mathcal{U}'$ is a CwF morphism $\mathcal{C}^{\mathcal{U}} \to \mathcal{C}^{\mathcal{U}'}$ that makes the following diagram commutative:

$$\begin{tikzcd}
\mathcal{C}^{\mathcal{U}} \arrow[rr] \arrow[dr] & & \mathcal{C}^{\mathcal{U}'} \arrow[dl] \\
& \mathcal{C}. &
\end{tikzcd}$$

If $\mathcal{C}$ is equipped with a Φ-structure ϕ, it is not possible in general to restrict ϕ to $\mathcal{C}^{\mathcal{U}}$. This justifies the following definition.

Definition 2.8.2. Let $\mathcal{C}$ be a Φ-category, where Φ is any type former, and $(\mathcal{U}, \mathsf{El})$ a universe in $\mathcal{C}$. We say that $\mathcal{U}$ is a Φ-universe if $\mathcal{C}^{\mathcal{U}}$ has a Φ-structure ϕ^{U} such that the canonical map $\mathcal{C}^{\mathcal{U}} \to \mathcal{C}$ is a Φ-morphism.

Note that if Φ is flat, then universes over $\mathcal{C}$ form a category, with morphisms given by universe morphisms such that the underlying CwF morphism preserves Φ-structures.

Definition 2.8.3. Let $\mathcal{A}$ be a category, Φ a flat type former and $\mathcal{C}$ a Φ-CwF. A *system of* Φ*-universes* on $\mathcal{C}$ (indexed by $\mathcal{A}$) is a functor from $\mathcal{A}$ to the category of Φ-universes of $\mathcal{C}$.

Usually, $\mathcal{A}$ is taken to be a poset, most commonly the ordinal ω. This is the case, for example, in the type theory described in [36].

Lemma 2.8.4. *In any RF-category $\mathcal{C}$, finite diagrams of fibrant objects have a limit.*

Proof. By lemma 2.1.42, all we have to prove is that any morphism between fibrant objects of $\mathcal{C}$ is isomorphic to a fibration. The following argument appears in [13].

Let A and B be types over the unit context, and $f : A \to B$ any map. Define:

$$E :\equiv (a : A) \times (b : B) \times (f(a) = b).$$

We have a factorisation:

$$\begin{tikzcd}
A \arrow[r, "i"] & E \arrow[r, "p"] & B,
\end{tikzcd}$$

and it is easy to see that i is an isomorphism, and p is a fibration. $\square$

Proposition 2.8.5. *Let $\mathcal{A}$ be a finite category and Φ a flat type former. There is a type former* $\text{UNIV}^{\Phi,\mathcal{A}}$ *such that systems of Φ-universes indexed by $\mathcal{A}$ are in bijective correspondence with* $\text{UNIV}^{\Phi,\mathcal{A}}$*-structures on $\mathcal{C}$.*

Proof. Define:

$$\Psi :\equiv (U : \mathcal{U}) \times (\mathsf{el} : \mathsf{El}\ U \to \mathcal{U}).$$

Clearly, a Ψ-structure is the same as a universe. Furthermore, $(\mathsf{El}[U], \mathsf{El}[\lambda^{-1}(\mathsf{el})])$ is a universe in RF_0/Ψ, which we will also denote with U. Therefore, RF_0/Ψ is an RF-category with universe U.

If $\mathcal{C}$ is a CwF equipped with a universe $\mathcal{V}$, we get an interpretation functor $[\![-]\!]^{\widehat{\mathcal{C}}} : RF_0/\Psi$ mapping U to Ty^V.

It follows that, if we define $\text{UNIV}^{\Phi,1}$ be the interpretation of Φ in RF_0/Ψ, a $\text{UNIV}^{\Phi,1}$-structure in $\mathcal{C}$ is the same as a Φ-universe $\mathcal{V}$ in $\mathcal{C}$.

Now, let $\mathcal{I}$ be the category with two objects 0 and 1, and only one non-identity morphism in $\mathcal{I}(0,1)$. Define a type former Ψ_2 as follows:

$$\begin{aligned}
\Psi_2 :\equiv\ & ((U, \mathsf{el}) : \Psi) \\
& \times ((U', \mathsf{el}' : \Psi)) \\
& \times (f_0 : \mathsf{El}\ U \to \mathsf{El}\ U') \\
& \times (f_1 : (X : \mathsf{El}\ U) \to \mathsf{El}(\mathsf{el}(X)) \to \mathsf{El}(\mathsf{el}'(f_0(X))).
\end{aligned}$$

Clearly, a Ψ_2-structure is the same as a pair of universes, together with a universe morphism, i.e. a system of universes indexed by $\mathcal{I}$.

Again, if $\mathcal{C}$ is equipped with universes $\mathcal{V}$ and $\mathcal{V}'$, there is an interpretation functor $[\![-]\!]^{\widehat{\mathcal{C}}} : RF_0/\Psi_2$ that maps the two universes U and U' in RF_0/Ψ_2 to V and V' respectively.

RF_0/Ψ_2 can be regarded as an RF-category, where the universe is defined to be:

$$(A : \mathcal{U}) \times (A' : \mathcal{U}') \times (f(A) = A').$$

Consequently, if we define $\text{UNIV}^{\Phi,\mathcal{I}}$ to be the interpretation of Φ in RF_0/Ψ_2, it is easy to see that a $\text{UNIV}^{\Phi,\mathcal{I}}$-structure on $\mathcal{C}$ is the same as a system of Φ-universes indexed by $\mathcal{I}$.

Now the general case follows from lemma 2.8.4 and the fact that every finite category is a finite colimit of 1 and $\mathcal{I}$ in Cat. □

2.8.1 *Univalent universes*

Let $\mathcal{C}$ be a Φ_0-CwF where Φ_0 is defined as:

$$\Phi_0 :\equiv \Phi^{\Pi} \times \Phi^{\Sigma} \times \Phi^{\text{IEQ}},$$

and let $(\mathcal{U}, \mathsf{El})$ be a universe in $\mathcal{C}$.

We can define the property of a function being an equivalence, internally in $\mathcal{C}$, as follows.

Over the context $(A, B : \mathcal{U})(f : A \to B)$, define a type $\mathsf{isEquiv}$:

$$\begin{aligned}\mathsf{isEquiv} :\equiv\ &((g : B \to A) \times (g \circ f = \mathrm{id})) \\ &\times ((g : B \to A) \times (f \circ g = \mathrm{id})).\end{aligned}$$

Here id and $\circ$ denote the identity function and composition of functions internal to $\mathcal{C}$, respectively, defined in the obvious way using the Π-type structure on $\mathcal{C}$.

The type Equiv of equivalences is defined over the context $(A, B : \mathcal{U})$:

$$\mathsf{Equiv} :\equiv (f : A \to B) \times \mathsf{isEquiv}[f].$$

It is easy to define a term $\mathrm{id}_E : \mathsf{Equiv}[A, A]$ over the context $(A : \mathcal{U})$, corresponding to the identity equivalence. From the properties of equality, it follows that there exists a function $\mathsf{coerce} : A = B \to \mathsf{Equiv}[A, B]$ in the context $(A, B : \mathcal{U})$.

Univalence for $\mathcal{U}$ is the following type, in the unit context:

$$\mathsf{ua}_{\mathcal{U}} :\equiv (A, B : \mathcal{U}) \to \mathsf{isEquiv}[A = B, \mathsf{Equiv}[A, B], \mathsf{coerce}[A, B]].$$

Definition 2.8.6. The universe $\mathcal{U}$ is said to be *univalent* if the corresponding univalence type $\mathsf{ua}_{\mathcal{U}}$ has a global element.

Proposition 2.8.7. *There is a type former Φ^{UA} over Φ, such that a Φ^{UA}-structure over Φ_0-CwF $\mathcal{C}$ is the same as a univalent universe.*

Proof. Univalence can be defined internally in any Φ_0-CwF, hence in particular in RF_0/Φ_0. □

2.9 FURTHER WORK

The definitions of special type formers given in section 2.7 serve their purpose of allowing a workable theory of type formers to be developed, but could be considered rather unsatisfactory, since they involve quantification over arbitrary functors, and it is thus hard to verify in practice that a given type former possesses those properties.

It seems reasonable that, at least for the case of *flat* and *algebraic* type formers, one should be able to verify that a type formers falls in one of those classes simply by inspecting the type expression in RF_0 that defines it.

For example, it appears to be the case that if a type former is written only using "first-order" Π-types of non-small types, then it is automatically flat. All the usual type formers, at least the ones that we used or mentioned, have this form, and the example of non-flat type former given in section 2.6 is indeed higher order.

It also seem likely that there should exist a notion of “strict positivity” for type formers, and those type formers that turn out to be strictly positive ought to be algebraic.

Investigating these and similar syntactic characterisations for type formers will be the goal of future research.

3

TWO-LEVEL TYPE THEORY

In this chapter, we will develop the idea of *two-level* type theory, modelled by CwFs with two type functors. Such systems are motivated by the need to introduce an internalised notion of *strict equality* into the theory.

Since certain type formers will play a special role within a two-level CwF, we single out CwFs with a fixed basic structure:

Definition 3.0.1. A *model of type theory* is a CwF equipped with Π, Σ and unit type structures. Given models of type theory $\mathcal{C}$ and $\mathcal{D}$, a morphism between them is a CwF morphism that preserves the Π, Σ and unit type structures.

We will write $\mathcal{T}$ to denote the type former corresponding to Π, Σ and unit types, so that a model of type theory is simply a CwF with a $\mathcal{T}$-structure. In other words:

$$\mathcal{T} :\equiv \Phi^{\Pi} \times \Phi^{\Sigma} \times \Phi^{\text{UNIT}}.$$

Example 3.0.2. If $\mathcal{C}$ is an arbitrary category, the presheaf category $\widehat{\mathcal{C}}$ is a model of type theory.

We will often simply say *model* instead of *model of type theory*. In particular, the structure needed to make a category (or a CwF) into a model will often be referred to as a *model structure*. Note that our notion of model structure is completely unrelated to that of *Quillen model structure* [30]. No confusion is possible, however, since we will never refer to the latter.

If Φ is a type former over $\mathcal{T}$, CwFs equipped with a Φ-structure will be referred to as Φ-models. If Φ is flat, Φ-models of type theory form a category $\mathcal{M}^{\Phi}$. In particular, the trivial type former over $\mathcal{T}$ is flat, and its corresponding category of models will be denoted simply by $\mathcal{M}$.

To incorporate strict equality into type theory, we will need to make a distinction between arbitrary types, and types for which weak equality is well defined. This is necessary, because, as we will see in lemma 4.2.1, the theory becomes degenerate if we don't make this distinction.

Definition 3.0.3. A *two-level CwF* is a CwF $\mathcal{C}$, equipped with a functor $\text{Ty}^{\text{f}} : \mathcal{C} \to \mathsf{Set}^{\text{op}}$, and a natural transformation $|-| : \text{Ty}^{\text{f}} \to \text{Ty}$.

Given a two-level CwF $\mathcal{C}$, we can define a second CwF structure on $\mathcal{C}$ having Ty^{f} as the type functor, and where terms are given by $\mathrm{Tm}^{\mathrm{f}}_{\Gamma}(A) :\equiv \mathrm{Tm}_{\Gamma}(|A|)$. Context extension is similarly defined as $\Gamma.A :\equiv \Gamma.|A|$. We will write $\mathcal{C}^{\mathrm{f}}$ to denote $\mathcal{C}$ equipped with this second CwF structure. To avoid confusion, and for consistency with notations that we will introduce later, we will write $\mathcal{C}^{\mathrm{s}}$, Ty^{s} and Tm^{s} when referring to the original CwF structure on $\mathcal{C}$.

The natural transformation $| - |$ induces a split CwF morphism $\mathcal{C}^{\mathrm{s}} \to \mathcal{C}^{\mathrm{f}}$.

Definition 3.0.4. A *two-level model of type theory* is a two-level CwF $\mathcal{C}$ such that both $\mathcal{C}^{\mathrm{s}}$ and $\mathcal{C}^{\mathrm{f}}$ are models of type theory, and $| - |$ is a $\mathcal{T}$-morphism.

If Φ and Ψ are type formers over $\mathcal{T}$, we define a (Φ, Ψ)-model to be a two-level model $\mathcal{C}$ where $\mathcal{C}^{\mathrm{f}}$ is equipped with a Φ-structure, and $\mathcal{C}^{\mathrm{s}}$ is equipped with a Ψ-structure.

The simplest way to construct a two-level CwF is with a universe:

Remark 3.0.5. Let $\mathcal{C}$ be a CwF equipped with a universe $\mathcal{U}$, El. Define $\mathrm{Ty}^{\mathrm{f}}(\Gamma) :\equiv \mathrm{Tm}_{\Gamma}(\mathcal{U})$, and for $A : \mathrm{Ty}^{\mathrm{f}}(\Gamma)$, let $|A| :\equiv \mathsf{El}[A]$.

Then $\mathcal{C}$, with the above choice of fibrant type functor, is a two-level CwF.

For all CwFs $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is a two-level CwF, where we can use the fibrant universe to define fibrant types as in remark 3.0.5.

3.1 THE SIMPLICIAL MODEL

The reference example of a two-level model is given by the category of *simplicial sets*, whose definition we recall below.

Definition 3.1.1. The *simplicial category* Δ has the natural numbers as objects, and morphisms $\Delta(n, m)$ are defined to be monotone functions $[n] \to [m]$, where $[k]$ denotes the set of natural numbers less or equal to k.

Definition 3.1.2. A *simplicial set* is a presheaf on Δ.

Simplicial sets form a category sSet, that can be regarded as a model of type theory like any presheaf category (example 3.0.2).

We can then define two-level model structure on sSet as follows: for all contexts Γ, fibrant types $\mathrm{Ty}^{\mathrm{f}}(\Gamma)$ are defined to be the subset of $\mathrm{Ty}(\Gamma)$ of those types A such that the display map $\Gamma.A \to \Gamma$ is a *Kan fibration*. Since Kan fibrations are closed under Π and Σ type formation [21], the fibrant fragment of sSet admits Π and Σ type formers, hence sSet is a two-level model of type theory.

Note that the definition of types used in [21] differs from the one we have given here. However, it can be easily verified that all the constructions carry over to our definition. Following [21], then, it can be shown that the fibrant fragment of sSet models all of the commonly used type formers, including a univalent universe.

3.2 PRESHEAF MODELS

In this section, we will show that, given a model of type theory $\mathcal{C}$, its presheaf category $\widehat{\mathcal{C}}$ can be regarded as a two-level model. Furthermore, if $\mathcal{C}$ is equipped with a $\mathbf{\Phi}$-structure for some type former $\mathbf{\Phi}$, one can find the same $\mathbf{\Phi}$-structure on the fibrant fragment of $\widehat{\mathcal{C}}$.

The idea of the proof is very simple: we start with a model $\mathcal{C}$ and build a two-level model structure on $\widehat{\mathcal{C}}$. The strict fragment of $\widehat{\mathcal{C}}$ is obtained from the usual CwF structure on presheaf categories (section 2.1.2). Fibrant types on $\widehat{\mathcal{C}}$ are given by the fibrant universe (section 2.1.3), and the fibrant model structure is inherited from that of $\mathcal{C}$.

The problem with this approach is that the resulting morphism from the fibrant to the strict fragment does not preserve type formers strictly. For example, let $A, B : \widehat{\mathcal{C}}(1, \mathrm{Ty})$ be fibrant types over the unit context. If we form their Σ-type within the fibrant model structure, then convert it to a strict type, we get the presheaf P given by:

$$P_\Gamma = \mathrm{Tm}_\Gamma(\Sigma_A B).$$

However, if we convert both A and B to strict types first, then take their Σ-type, we end up with a presheaf Q, where, Q_Γ is a set of pairs of terms of A and B over Γ.

Of course, terms of $\Sigma_A B$ can be identified to the set of such pairs, but the two resulting presheaves, although isomorphic, are not equal on the nose. A similar problem occurs with Π-types. Therefore, the resulting structure on $\widehat{\mathcal{C}}$ does not satisfy the definition of two-level model (definition 3.0.4).

For this reason, we need to slightly modify the CwF structure on $\mathcal{C}$, so that strict preservation of Π and Σ can be achieved. This will be the aim of the following subsections.

3.2.1 *Lifting type formers*

Let $\mathcal{C}$ be a CwF equipped with a $\mathbf{\Phi}$-structure ϕ. The fibrant universe Ty determines a CwF structure on $\widehat{\mathcal{C}}$, where types are given by

$$\widehat{\mathrm{Ty}}^{\mathrm{f}}(P) :\equiv \widehat{\mathcal{C}}(P, \mathrm{Ty}).$$

Let us denote by $\widehat{\mathcal{C}}^{\mathrm{f}}$ the corresponding CwF.

Lemma 3.2.1. *The yoneda embedding $y : \mathcal{C} \to \widehat{\mathcal{C}}^{\mathrm{f}}$ can be extended to a CwF morphism.*

Proof. By the Yoneda lemma, $\mathrm{Ty}^{\mathrm{f}}(y\Gamma) \cong \mathrm{Ty}(\Gamma)$, hence we can take this isomorphism as the action of y on types. Consequently, y can be defined to be an isomorphism on terms as well. □

In this section, we will show how to lift ϕ to a Φ-structure on $\widehat{\mathrm{Ty}}^{\mathrm{f}}$, so that the Yoneda embedding of lemma 3.2.1 is a Φ-morphism.

Lemma 3.2.2. *Let $f : \mathcal{C} \to \mathcal{D}$ be a CwF equivalence between RF-categories. If f weakly preserves the universe, then f is a weak RF-morphism.*

Proof. Since f is a CwF equivalence, we can use it to transport all the type structures from $\mathcal{C}$ to $\mathcal{D}$. Since all the type structures of an RF-category except the universe are characterised by a universal property, it easily follows that the transported structures are isomorphic to the original ones on $\mathcal{D}$, which amounts to saying that f preserves them. □

Theorem 3.2.3. *Let $f : \mathcal{C} \to \mathcal{D}$ be a CwF morphism such that $f^* : \widehat{\mathcal{D}} \to \widehat{\mathcal{C}}$ is an equivalence of categories, and f is bijective on types. Suppose $\mathcal{C}$ is equipped with a Φ-structure ϕ. Then there exists a Φ-structure ϕ' on $\mathcal{D}$ such that f is a Φ-morphism.*

Proof. Since f^* is an equivalence of categories, it induces a equivalences of slice categories, hence an isomorphism of the type functors of $\widehat{\mathcal{D}}$ and $\widehat{\mathcal{C}}$ thanks to proposition 2.1.7. Therefore, f^* is a CwF equivalence.

Note that f being bijective on types is equivalent to f^* preserving the universe. Hence, it follows from lemma 3.2.2 that f^* is a weak RF-morphism.

Let $f_! : \widehat{\mathcal{C}} \to \widehat{\mathcal{D}}$ be the left adjoint of f^*. Explicitly, $f_!$ is given by the left Kan extension of $y \circ f$ along $y : \mathcal{C} \to \widehat{\mathcal{C}}$. In this case, $f_!$ is also a CwF equivalence, hence a weak RF-morphism.

We define a CwF morphism $\widetilde{f} : \widehat{\mathcal{C}} \to \mathcal{R}_{f^*}$. On objects, $\widetilde{f}$ maps P to the triple $(P, f_!(P), \delta_P)$, where δ_P denotes the span:

$$\begin{tikzcd} f^* f_! P & P \arrow[l, "\eta"'] \arrow[r] & P. \end{tikzcd}$$

The action of $\widetilde{f}$ on types is defined similarly. It is not hard to check that $\widetilde{f}$ is a weak RF-morphism.

Therefore, the diagram:

$$\begin{tikzcd} & RF_0 \arrow[dl] \arrow[dr] & \\ \widehat{\mathcal{C}} \arrow[rr, "\widetilde{f}"'] & & \mathcal{R}_f \end{tikzcd}$$

commutes weakly by theorem 2.2.5. It follows that $\widetilde{f}(\phi)$ determines a canonical Φ-structure on $\widehat{\mathcal{D}}$, such that f is a Φ-morphism, as required. □

Theorem 3.2.4. *There is a Φ-structure on $\widehat{\mathcal{C}}^{\mathrm{f}}$ such that the Yoneda embedding (lemma 3.2.1) preserves Φ-structures.*

Proof. The Yoneda embedding $y : C \to \widehat{\mathcal{C}}^{\mathrm{f}}$ is bijective on types, and the induced functor y^* is an equivalence. Therefore, theorem 3.2.3 applies directly. □

3.2.2 *Regular models*

Let $\mathcal{C}$ be a model, and consider the category $\widehat{\mathcal{C}}/\mathrm{Ty}$ of presheaves over Ty. If X is such a presheaf, we denote by $|-|_X$ the corresponding morphism to Ty.

For a presheaf X over Ty, regard X as a type in the unit context of the CwF $\widehat{\mathcal{C}}$, and denote by $X^{(2)}_\Gamma$ the presheaf corresponding to the type:

$$\Sigma_{A:X}\Pi_{\mathrm{Tm}[|A|_X]}X.$$

Lemma 3.2.5. *For all context $\Gamma : \mathcal{C}$, the set $X^{(2)}_\Gamma$ is naturally isomorphic to the set of pairs (A, B), where $A : X_\Gamma$ and $B : X_{\Gamma.|A|_X}$.*

Proof. Immediate consequence of lemma 2.1.16. □

In the following, we will use the isomorphic representation of $X^{(2)}$ given by lemma 3.2.5 liberally.

Note that if Ty is regarded as an element of $\widehat{\mathcal{C}}/\mathrm{Ty}$, the presheaf $\mathrm{Ty}^{(2)}$ matches with the one we defined in section 2.1.3.

We can make $X^{(2)}$ into a presheaf over Ty in at least two ways: using the Π or Σ-type structures on $\mathcal{C}$. In fact, they both can be regarded as morphisms:

$$\mathrm{Ty}^{(2)} \to \mathrm{Ty},$$

from which we obtain the desired morphism $X^{(2)} \to \mathrm{Ty}$ by composing with the obvious map $X^{(2)} \to \mathrm{Ty}^{(2)}$.

We now define an endofunctor E of $\widehat{\mathcal{C}}/\mathrm{Ty}$ as:

$$EX :\equiv X^{(2)} + X^{(2)} + 1,$$

where the map $EX \to \mathrm{Ty}$ on the first $X^{(2)}$ component is given by the Π-type structure on $\mathcal{C}$ as explained above, on the second component by the Σ-type structure, and on the third component it just selects the unit type.

Denote by:

$$\pi^E : X^{(2)} \to EX$$
$$\sigma^E : X^{(2)} \to EX$$
$$u^E : 1 \to EX$$

the three canonical injections into the coproduct EX.

Proposition 3.2.6. *The endofunctor $E : \widehat{\mathcal{C}}/\mathrm{Ty} \to \widehat{\mathcal{C}}/\mathrm{Ty}$ is finitary.*

Proof. Clear from the characterisation of lemma 3.2.5. □

It follows from proposition 3.2.6 that E admits a free monad E^*.

An element of $(E^*X)_\Gamma$ is either a *base element* $\eta(A)$, where $A : X_\Gamma$ and $\eta : X \to E^*X$ is the unit of the monad E^*, or a *compound element* of the form $\pi^E(A, B)$, $\sigma^E(A, B)$ or u^E.

Here, we are abusing notation by writing π^E for the canonical map $(EX)^{(2)} \to EX$ given by the free monad construction, and similarly for σ^E and u^E.

The idea of this construction becomes clear when we try to apply E^* to Ty itself. The resulting presheaf $E^*\mathrm{Ty}$ can be regarded as an alternative type functor on $\mathcal{C}$ where types can be uniformly be classified into base types, Π-types, Σ-types or unit types.

This is made precise by the following.

Lemma 3.2.7. *For any model $\mathcal{C}$, the presheaf $E^*\mathrm{Ty}$ can be extended to a model structure.*

Proof. For an element $A : E^*\mathrm{Ty}$, define its set of terms simply as $\mathrm{Tm}_\Gamma(|A|)$. This clearly equips $\mathcal{C}$ with a CwF structure, where context extension is given by $\Gamma.A :\equiv \Gamma.|A|$.

The rest of the structure can be obtained directly from the decomposition of E^*: the Π-type structure is given by π^E, the Σ-type structure by σ^E and the unit type structure by u^E. Verifying all the required properties is straightforward. $\square$

If $\mathcal{C}$ is a category and Φ is a flat type former over $\mathcal{T}$, write $\mathcal{M}^\Phi_\mathcal{C}$ for the subcategory of $\mathcal{M}^\Phi$ consisting of Φ-models that have $\mathcal{C}$ as the underlying category, and Φ-morphisms that have the identity as the underlying functor. We refer to $\mathcal{M}^\Phi_\mathcal{C}$ as the *category of Φ-model structures on $\mathcal{C}$.*

Similarly, $\mathcal{M}_\mathcal{C}$ denotes the category of model structures on $\mathcal{C}$ (without any additional structure).

Lemma 3.2.7 implies that E^* induces an endofunctor on $\mathcal{M}_\mathcal{C}$ for all models $\mathcal{C}$.

Lemma 3.2.8. *The endofunctor determined by E^* is a comonad on $\mathcal{M}_\mathcal{C}$.*

Proof. A morphism ϵ, serving as the counit of the comonad, can be obtained directly from the map $| - | : E^*\mathrm{Ty} \to \mathrm{Ty}$. All we need to do to make ϵ into the unit of a comonad is to show that it induces a model morphism. Indeed, this is readily verified, since the model structure corresponding to $E^*\mathrm{Ty}$ is defined in terms of $| - |$ itself.

To define the comonad multiplication $\delta : E^*\mathrm{Ty} \to E^*(E^*\mathrm{Ty})$, we proceed by induction on the structure of E^*, and at the same time show that $|\delta(X)| = |X|$ for all X.

Let $\Gamma : \mathcal{C}$, and $X : E^*\mathrm{Ty}(\Gamma)$.

- If $X = \eta(A)$ for some $A : \mathrm{Ty}(\Gamma)$, set $\delta(X) :\equiv \eta(\eta(A))$. Then clearly $|\delta(X)| = A = |X|$.
- If $X = \pi^E(A, B)$, we have by induction hypothesis $\delta(A) : \mathcal{E}^*(\mathcal{E}^*\mathrm{Ty})(\Gamma)$, and, modulo an application of the isomorphism of lemma 3.2.5, $\delta(B) : \mathcal{E}^*(\mathcal{E}^*\mathrm{Ty})(\Gamma.|A|)$. Since $|A| = |\delta(A)|$ by the induction hypothesis, we can set $\delta(X) :\equiv \pi^E(\delta(A), \delta(B))$, and observe that $|\delta(X)| = \Pi_{|A|}|B| = |\pi^E(A, B)| = |X|$, as required.
- If $X = \sigma^E(A, B)$, we proceed exactly like for the π^E case above.
- If $X = u^E$, we set $\delta(X) :\equiv u^E$, and the required equation obviously holds.

The fact that δ is a model morphism follows immediately from its definition.

One of the comonad laws has already been proved as part of the definition of δ, and the others can be easily verified. □

Lemma 3.2.8 may seem surprising at first, since E^* is defined as a (free) monad, while it turns out to be a comonad when regarded as an endofunctor of model structures. However, E^* is already a comonad on $\widehat{\mathcal{C}}/\mathrm{Ty}$, so all that lemma 3.2.8 states is that this structure carries over.

On the other hand, E^* is *not* a monad on $\mathcal{M}_{\mathcal{C}}$, since for example the unit $\eta : \mathsf{Id} \to E^*$ cannot be regarded as a model morphism.

Theorem 3.2.9. *Let $\mathcal{C}$ be a model. The Yoneda embedding $y : \mathcal{C} \to \widehat{\mathcal{C}}$ can be extended to a model morphism between $E^*\mathrm{Ty}$ and the canonical model structure on $\widehat{\mathcal{C}}$ defined in section 2.1.2.*

Proof. We have already defined an action of y on types, denoted $y_0 : \mathrm{Ty} \to \widehat{\mathrm{Ty}}(y-)$. The function y_0 maps a type $A : \mathrm{Ty}(\Gamma)$ to the functor $(\Delta, \sigma) \mapsto \mathrm{Tm}_\Delta(A[\sigma])$ (definition 2.1.35). We observed that y_0 is not in general a model morphism.

For an element $X : E^*\mathrm{Ty}(\Gamma)$, we will define $y(X) : \widehat{\mathrm{Ty}}(y\Gamma)$ by induction on the structure of $E^*\mathrm{Ty}(\Gamma)$, and at the same time we will construct a natural isomorphism $y(X) \cong y_0(|X|)$.

- if $X = \eta(A)$ for some type $A : \mathrm{Ty}(\Gamma)$, let $y(X) :\equiv y_0(X)$ and the isomorphism be the identity;
- if $X = \pi^E(A, B)$, we get by induction hypothesis a type $y(A) : \widehat{\mathrm{Ty}}(y\Gamma)$; similarly, using lemma 3.2.5, we get a type $y'(B) : \widehat{\mathrm{Ty}}(y(\Gamma.A))$. Now, $y(\Gamma.A) \cong y(\Gamma).y_0(A) \cong y(\Gamma).y(A)$, hence we can set $y(X) :\equiv \Pi_{y(A)} y'(B)$. It follows from the definition of Π-types in $\mathcal{C}$ that $y(X) \cong y_0(|X|)$.
- if $X = \sigma^E(A, B)$ or $X = u^E$, we proceed similarly to the case of π^E.

We can prove that $y : \mathrm{Ty}(\Gamma) \to \widehat{\mathrm{Ty}}(y\Gamma)$ is natural in Γ by induction on its argument, and using the fact that y_0 is natural (proposition 2.1.38) as the base case.

At this point, since the definition of the Π-type structure on $E^*\mathrm{Ty}$ is given precisely by π^E, it is easy to verify that y as defined above does indeed preserve Π-types strictly, and a similar argument shows that y preserves all type formers, hence it is a model morphism. □

Definition 3.2.10. A Φ-model $\mathcal{C}$ is said to be *(weakly) regular* if it is equipped with a model morphism $\theta : \mathrm{Ty} \to E^*\mathrm{Ty}$ on the category of Φ-model structures of $\mathcal{C}$. $\mathcal{C}$ is said to be *strongly regular* if θ is a coalgebra of the comonad E^*.

Definition 3.2.11. An type former Φ over $\mathcal{T}$ is said to be *regular* if E^* maps Φ-models into Φ-models.

If Φ is flat, we can say that Φ is regular if and only if E^* can be extended to an endofunctor (hence a comonad) on $\mathcal{M}^{\Phi}_{\mathcal{C}}$ for any Φ-model $\mathcal{C}$. Clearly, the trivial type former over $\mathcal{T}$ is regular by lemma 3.2.7.

Proposition 3.2.12. *If Φ a regular algebraic type former over $\mathcal{T}$, then the initial Φ-model is strongly regular.*

Proof. Let $\mathcal{S}$ be the initial Φ-model. The existence of $\theta : \mathrm{Ty}^{\mathcal{S}} \to E^*\mathrm{Ty}^{\mathcal{S}}$ is an immediate consequence of the initiality of $\mathcal{S}$. □

Theorem 3.2.13. *Let Φ and Ψ be type formers over $\mathcal{T}$, with Φ regular and Ψ set theoretic, and let $\mathcal{C}$ be a regular model. The presheaf category $\widehat{\mathcal{C}}$ can be equipped with a (Φ, Ψ)-model structure such that the Yoneda embedding $y : \mathcal{C} \to \widehat{\mathcal{C}}$ can be extended to a Φ-morphism between $\mathcal{C}$ and the fibrant fragment of $\widehat{\mathcal{C}}$.*

Proof. Let $\mathcal{C}'$ be the model obtained from $\mathcal{C}$ by replacing Ty with $E^*\mathrm{Ty}$. We know from theorem 3.2.4 that $\widehat{\mathcal{C}}^{\mathrm{f}}$ can be made into a Φ-model and $y : \mathcal{C}' \to \widehat{\mathcal{C}}^{\mathrm{f}}$ can be extended to a Φ-morphism. Furthermore, $\widehat{\mathcal{C}}^{\mathrm{s}}$ is a Ψ-model by the assumption that Ψ is set-theoretic.

Since $\theta : \mathcal{C} \to \mathcal{C}'$ is a model morphism, all we have to do is define the rest of the two-level model structure on $\widehat{\mathcal{C}}$.

The non-obvious bit is how to define the coercion map $| - | : \widehat{\mathrm{Ty}}^{\mathrm{f}} \to \widehat{\mathrm{Ty}}^{\mathrm{s}}$. Fortunately, most of the hard work is already contained in the proof of theorem 3.2.9.

For all contexts $P : \widehat{\mathcal{C}}$, and $A : \widehat{\mathrm{Ty}}(P)$, set:

$$|A|_{\Gamma}(x) :\equiv y(A_{\Gamma}(x))_{\Gamma}(\mathrm{id}).$$

From naturality of y, it follows that:

$$|A|[x] = y(A_{\Gamma}(x)).$$

Now, consider a pair (A, B) in the fibrant fragment. Its related pair is given (lemma 2.1.29) by $(|A|, |B|)$, where we have used the isomorphism of corollary 2.1.17 implicitly.

Now we compute:

$$\begin{aligned}
|\Pi_A B|_\Gamma(x) &= y((\Pi_A B)_\Gamma(x))_\Gamma(\mathrm{id}) \\
&= y(\Pi_{A_\Gamma(x)} \widetilde{B}_\Gamma(x))_\Gamma(\mathrm{id}) \\
&= (\Pi_{y(A_\Gamma(x)} y'(\widetilde{B}_\Gamma(x)))_\Gamma(\mathrm{id}) \\
&= (\Pi_{|A|[x]} |B|[x^+])_\Gamma(\mathrm{id}) \\
&= ((\Pi_{|A|} |B|)[x])_\Gamma(\mathrm{id}) \\
&= (\Pi_{|A|} |B|)_\Gamma(x).
\end{aligned}$$

It follows that $| - |$ preserves Π types strictly. A similar verification for Σ and the unit type shows that $| - |$ is a model morphism, concluding the proof. □

3.2.3 *Conservativity*

An important consequence of the results of section 3.2.2 is the following *conservativity* result.

Theorem 3.2.14. *Let Φ be a regular algebraic type former, and Ψ a set-theoretic type former over $\mathcal{T}$. Assume that the category of (Φ, Ψ)-models has an initial object $\mathcal{S}$, and let $\mathcal{H}$ be the initial Φ-model. Let $H : \mathcal{H} \to \mathcal{S}$ be the unique morphism to the fibrant fragment of $\mathcal{S}$.*

Let $\Gamma : \mathcal{H}$ and $A : \mathrm{Ty}(\Gamma)$. If $H(A)$ is inhabited in $\mathcal{S}$, then A is inhabited in $\mathcal{H}$.

Proof. Consider the diagram:

$$\begin{tikzcd}
\mathcal{H} \arrow[rr, "y"] \arrow[dr, "H"'] & & \widehat{\mathcal{H}} \\
& \mathcal{S}, \arrow[ur] &
\end{tikzcd}$$

where $\widehat{\mathcal{H}}$ is regarded as a (Φ, Ψ)-model as in theorem 3.2.13.

Since $\mathcal{H}$ is the initial Φ-model, this diagram commutes weakly by proposition 2.7.6.

Therefore, if $H(A)$ is inhabited in $\mathcal{S}$, it is also inhabited in $\widehat{\mathcal{H}}$, hence in $\mathcal{H}$, since the Yoneda embedding is full. □

Theorem theorem 3.2.14 states that to prove a proposition or construct a value in a model, it is enough to prove it or construct it in a corresponding two-level model. The type formers Φ and Ψ appearing in theorem 3.2.14 specify the choice of structure for the fibrant fragment and strict fragment of the two level model, respectively. They are both type formers over $\mathcal{T}$, because they share the common structure of a model of type theory, which, according to definition 3.0.4, has to be preserved by the coercion morphism from fibrant to strict types.

Note that individual type formers outside of the common fragment in $\mathcal{T}$ may be duplicated across $\mathbf{\Phi}$ and $\mathbf{\Psi}$. For example, both $\mathbf{\Phi}$ and $\mathbf{\Psi}$ could contain the type former for binary sums $\mathbf{\Phi}^{\mathrm{SUM}}$ introduced in section 2.4. This is not a problem, but it is important to note that the common type formers outside of $\mathcal{T}$ need not be preserved by the coercion morphism.

Regularity of $\mathbf{\Phi}$ is important, because without it we cannot make sure that preservation of the basic type former $\mathcal{T}$ is strict. It could be possible to define a weaker notion of two-level model of type theory that, unlike definition 3.0.4, does not require the basic type formers to be preserved strictly by the coercion morphism. In that case, it would be possible to remove the regularity assumption from the hypotheses of theorem 3.2.14.

The intended application of theorem 3.2.14 is to a setting where $\mathbf{\Phi}$ contains the type formers of a theory like HoTT, and $\mathbf{\Psi}$ the ones for a version of strict type theory, either something like our *RF*, or alternatively a theory with just UIP and function extensionality (section 2.4). In section 4.1 we will describe such a setting in detail.

3.3 TWO-LEVEL TYPE FORMERS

When building a two-level system, one can specify type formers $\mathbf{\Phi}$ and $\mathbf{\Psi}$ over $\mathcal{T}$, and that gives a notion of $(\mathbf{\Phi}, \mathbf{\Psi})$-model that one can work with.

This way, the type formers of $\mathbf{\Phi}$ and $\mathbf{\Psi}$, except for their $\mathcal{T}$ fragment, are completely independent, which means that the strict and fibrant fragment of a $(\mathbf{\Phi}, \mathbf{\Psi})$-model do not interact outside of their common model of type theory.

Sometimes, however, it might be desirable to put structures on top of a two-level model that make full use of the two fragments. To make this possible, we will define a notion of *two-level type former*.

Definition 3.3.1. A *two-level RF-category* is an *RF*-category with an additional universe $(\mathcal{U}^{\mathrm{f}}, \mathsf{El}^{\mathrm{f}})$, and a morphism of universes $\mathcal{U}^{\mathrm{f}} \to \mathcal{U}$.

To avoid confusion, we will denote the first universe in an *RF*-category with $(\mathcal{U}^{\mathrm{s}}, \mathsf{El}^{\mathrm{s}})$. We will often keep the morphism $\mathcal{U}^{\mathrm{f}} \to \mathcal{U}^{\mathrm{s}}$ implicit when writing out types and terms in a two-level *RF*-category.

Similarly to what we did in section 2.3, we can define a category $\mathcal{RF}^2$ of two-level *RF*-categories, and show that it has an initial object RF_0^2.

Consequently, we get the corresponding notions of *two-level type former* and *two-level structure* for a two-level CwF.

Furthermore, there is a two-level type former $\mathcal{T}^2$ corresponding to the statements that both $\mathcal{U}^{\mathrm{f}}$ and $\mathcal{U}^{\mathrm{s}}$ are $\mathcal{T}$-universes, and that the map $\mathcal{U}^{\mathrm{f}} \to \mathcal{U}^{\mathrm{s}}$ is a $\mathcal{T}$-universe morphism.

Correspondingly, for a two-level type former Φ over $\mathcal{T}^2$, we get a corresponding notion of two-level Φ-model.

Note that a type former Φ can be regarded as a two-level type former in two ways, either by lifting it to $\mathcal{U}^{\mathrm{f}}$ or $\mathcal{U}^{\mathrm{s}}$. If Φ is over $\mathcal{T}$, then either of its liftings to RF_0^2 are over $\mathcal{T}^2$.

In particular, given type formers Φ and Ψ over $\mathcal{T}$, we can lift Φ to a two-level type former Φ^{f} on $\mathcal{U}^{\mathrm{f}}$, Ψ to a two-level type former Ψ^{s} on $\mathcal{U}^{\mathrm{s}}$, and obtain a two-level type former $\Phi^{\mathrm{f}} \times \Psi^{\mathrm{s}}$ over $\mathcal{T}^2$. Then (Φ, Ψ)-models are the same as two-level $\Phi^{\mathrm{f}} \times \Psi^{\mathrm{s}}$-models.

We can then prove a more general version of theorem 3.2.14 for models of two-level type formers.

Definition 3.3.2. Let Φ be a type former over $\mathcal{T}$, and Ψ a two-level type former over $\Phi^{\mathrm{f}} \times_T \mathcal{T}^2$. We say that Ψ is *set-theoretic* if for all regular Φ-models $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}}$ is a two-level Ψ-model.

Theorem 3.3.3. *Let Φ be a regular algebraic type former over $\mathcal{T}$, and Ψ a set-theoretic two-level type former over $\Phi \times_{\mathcal{T}} \mathcal{T}^2$. Assume that the category of two-level Ψ-models has an initial object $\mathcal{S}$, and let $\mathcal{H}$ be the initial Φ-model. Let $H : \mathcal{H} \to \mathcal{S}$ be the unique morphism to the fibrant fragment of $\mathcal{S}$.*

Let $\Gamma : \mathcal{H}$ and $A : \mathrm{Ty}(\Gamma)$. If $H(A)$ is inhabited in $\mathcal{S}$, then A is inhabited in $\mathcal{H}$.

Proof. Completely analogous to the proof of theorem 3.2.14. $\square$

4

TYPE THEORY WITH STRICT EQUALITY

In this chapter, we fix a specific two-level model of type theory, and work internally in it. One is free to assume that this model is the initial one equipped with the prescribed type structures, but this is not strictly necessary, so we will not make that assumption.

The style used in the following mimics that employed in [36] to develop HoTT *internally*. We will make use of the same ideas, although our notation is consistent with the rest of the thesis, and follows the conventions described in sections 2.1.1 and 2.1.7.

Our main purpose for this chapter is to develop enough fundamentals of two-level type theory to be able to define certain basic notions that will enable us to express the idea of "infinite structure" or "infinite tower of coherence conditions", as explained in section 1.6.

4.1 INTRODUCTION

Let Φ_0 be a "basic" type former over $\mathcal{T}$. For concreteness, define Φ_0 as:

$$\Phi_0 :\equiv \Phi^{\text{IEQ}} \times \Phi^{\text{SUM}} \times \Phi^{\text{EMPTY}} \times \Phi^{\mathbb{N}}$$

since these are the type structures that will be assumed to exist in the following.

The type former Φ_0 represents structures that will be present both in the fibrant and in the strict fragment of our theory. However, since the definition of two-level model only requires the two $\mathcal{T}$-structures to be compatible, the two Φ_0-structures will behave very differently, in general.

For the rest of the chapter, fix a (Φ, Ψ) model of type theory $\mathcal{A}$, where:

- Ψ is the type former over Φ obtained by adding function extensionality (Φ^{FUNEXT}), and requiring that the equality in Φ satisfy UIP;
- the strict fragment of $\mathcal{A}$ admits a system of Ψ-universes indexed by some finite ordinal n;

- the fibrant fragment of $\mathcal{A}$ admits a system of *univalent* Φ-universes indexed by n;
- the two systems of universes can be extended to a system of universes indexed by $\omega \times 2$.

The idea of the universe setup is that the two systems of universes live in the two different fragments, but for any $i : n$, the i-th fibrant universe is "contained" in the i-th strict universe.

A crucial observation is that we can find a two-level type former Ψ' so that the initial two-level Ψ'-model satisfies all the above conditions, and at the same type all the hypotheses of theorem 3.3.3.

To make this possible, we have to set up the type formers for our universes so that all the fibrant universes are contained in the first strict one. This makes the two-level type former Ψ' for set-theoretic.

Therefore, if we assume $\mathcal{A}$ to coincide with the initial two-level Ψ'-model, we are allowed to interpret all the results of this chapter to ordinary HoTT, thanks to theorem 3.3.3.

As mentioned above, the type structures for the strict and the fibrant fragments are not required to match (outside of $\mathcal{T}$). However, it is possible to assume that parts of them do.

In particular, the language is (at least apparently) more expressive if we require the Φ^{SUM}, Φ^{EMPTY} and $\Phi^{\mathbb{N}}$-structures to match. A model where this happens has been referred to as *strong* in [4].

One substantial disadvantage of working in a strong model is that theorem 3.3.3 does not apply. It appears that strong two-level models constitute a proper extension of HoTT, which means that adopting their language implies having to depart from HoTT itself. Therefore, we will *not* make this assumption in the following.

4.1.1 *Differences with HTS*

Although our two-level theory is inspired by HTS [38], and shares many of its features and motivations, there are some substantial differences between the two systems.

Probably the most important difference is that HTS assumes that natural numbers, binary sums and the empty type in the fibrant fragment can eliminate to arbitrary types. In other words, coercion from fibrant to strict types preserves those type formers. As we observed above, the extra assumptions would break the proof of our conservativity result (theorem 3.2.14). Furthermore, they are not strictly necessary for the development that follows.

Another fundamental difference is that HTS assumes the reflection rule for equality in the strict fragment. From a semantic point of view, this is a completely unproblematic assumption, and in fact it is within the scope of theorem 3.2.14, since equality in presheaf categories does validate the reflection rule.

However, systems with equality reflection seem to be much harder to study from a meta-theoretical point of view, and consequently harder to implement. Although most of the current implementation efforts for proof assistants based on Martin-Löf type theory do not include equality reflection, there have been recent attempts at developing a system within which something like HTS could potentially be realised [6].

In practice, lack of a reflection rule for strict equality does not seem to be a big hurdle when reasoning within a two-level system *informally.* Of course, formalising proofs in a proof assistant could potentially be made easier by not having to manually manage rewrites along equality witnesses, but we have no reason to believe that a system that replaces reflection with simply UIP would be any less practical for actual formalisation of results based on a two-level theory.

Finally, universes in the strict fragment of our system are not assumed to be fibrant types, like in HTS. In some variations of HTS, universes of strict types are even assumed to be contractible. This is motivated by their interpretation in the simplicial set model (section 3.1). However, universes in presheaf categories are clearly not fibrant in the two-level CwF structure that we constructed in section 3.2, so we will not make this assumption.

4.2 BASIC NOTIONS

We will adopt some specific conventions when working internally in a two-level theory.

As in any two-level model of type theory, we have a distinction between fibrant and strict types. Technically, they are completely disjoint sets, only connected by the coercion morphism $\mathrm{Ty}^{\mathrm{f}} \to \mathrm{Ty}^{\mathrm{s}}$.

However, we will sometimes refer to *being fibrant* as a property of a strict type: such a type will be called fibrant if there is a fibrant type that coerces to it.

We will keep the universe hierarchies of the two fragments distinct, by writing $\mathcal{U}$ for a generic fibrant universe, and $\mathcal{U}^{\mathrm{s}}$ for a strict one. Similarly to how we dealt with universes in the metatheory (section 2.1.1), we will not write explicit subscripts to identify a universe within a hierarchy, and instead adhere to the convention called "typical ambiguity" [12].

Similarly, we will use the superscript s to denote type formers for the strict fragment, and no superscript at all for their fibrant counterparts. For example

0^{s} is the strict empty type, $A +^{\mathrm{s}} B$ is a strict binary sum, etc. For strict Π, Σ and unit types, we are free to omit the subscript, since they behave identically to their fibrant versions. Furthermore, strict equality will be written as $x \underset{s}{=} y$, and fibrant equality simply as $x = y$.

We will follow the same convention for defined notions. For example, we will write $A \simeq^{\mathrm{s}} B$ to denote *strict isomorphism*, defined as follows:

$$\begin{aligned} A \simeq^{\mathrm{s}} B :\equiv {} & (f : A \to B) \\ & \times (g : B \to A) \\ & \times ((a : A) \to g(fa) \underset{s}{=} a) \\ & \times ((b : B) \to f(gb) \underset{s}{=} b). \end{aligned}$$

Of particular importance for the following are the *finite ordinals* given by $\mathsf{Fin}_n : \mathcal{U}$. They are defined by induction on the natural number argument $n : \mathbb{N}$:

$$\begin{aligned} \mathsf{Fin}_0 &:\equiv 0 \\ \mathsf{Fin}_{n+1} &:\equiv 1 + \mathsf{Fin}_n. \end{aligned}$$

Of course, we also get the corresponding strict type $\mathsf{Fin}^{\mathrm{s}}_n$, indexed over the strict natural numbers, with the analogous strict definition.

We conclude this section with the following observation, showing that, in order to develop a system with two different notions of equality, one really needs the separation between fibrant and strict types.

Lemma 4.2.1. *Assume that the coercion morphism* $\mathrm{Ty}^{\mathrm{f}} \to \mathrm{Ty}^{\mathrm{s}}$ *is an isomorphism. Then strict and fibrant equality coincide up to equivalence, hence in particular fibrant equality satisfies* UIP.

Proof. For any type A, and $a, b : A$, it follows from the assumption that the strict equality type $a \underset{s}{=} b$ is fibrant. Therefore, we can define a function:

$$f : a \underset{s}{=} b \to a = b,$$

and it is easy to show that f is the inverse of the usual coercion $a = b \to a \underset{s}{=} b$. Therefore, strict and fibrant equality are strictly isomorphic types. □

4.3 FIBRANT REPLACEMENT

It is natural to ask whether we could extend our theory with a *fibrant replacement* operation, allowing us to convert any type into its "closest" fibrant approximation.

In fact, it is not hard to give a definition for a fibrant replacement type former in RF_0^2:

$$\begin{aligned}\Phi^R : (A : \mathcal{U}^{\mathrm{s}}) \to &(R : \mathcal{U}^{\mathrm{f}})\\ &\times(\eta : A \to R)\\ &\times(\mathsf{elim} : (X : \mathcal{U}^{\mathrm{f}}) \to (A \to X) \to R \to X)\\ &\times((X : \mathcal{U}^{\mathrm{f}})(f : A \to X)(a : A) \to \mathsf{elim}^X(f)(\eta(a)) = f(a))\end{aligned}$$

The type former Φ^R expressed quite faithfully the idea of “replacing” a strict type with a fibrant approximation: given a strict type A, we get a fibrant type R, together with a function $\eta : A \to R$, and a universal property stating that, for any fibrant type X, to define a function $R \to X$ all we need it to define a function $A \to X$.

In fact, a fibrant replacement type former is quite similar to the propositional truncation operation, only, of course, it makes types *fibrant* rather than propostional.

Having fibrant replacement in the theory would make a lot of constructions easier, and it does seem justifiable, since many of the known models of HoTT$^{\mathrm{s}}$, being Quillen model categories, are indeed equipped with a very similar operation.

For example, a type former along the lines of Φ^R is considered in [8], where the authors construct a model structure on a universe of strict types using fibrant replacement.

Unfortunately, it turns out that the fibrant replacement operation in models of HoTT$^{\mathrm{s}}$ cannot be internalised as a Φ^R-structure, as the following theorem shows.

Theorem 4.3.1. *Assume the existence of a fibrant replacement type structure R, as given by Φ^R. Then every fibrant type is a set.*

Proof. Let A be a fibrant type, and $x : A$. Since the type $(r : x \underset{s}{=} x) \to r = \mathsf{refl}$ is inhabited, so is its fibrant replacement. Therefore, by path induction, we get that for all $x, y : A$ and $p : x = y$:

$$R((r : x \underset{s}{=} y) \to r = p).$$

However, if $p : x = x$, the type $(r : x \underset{s}{=} x) \to r = p$ clearly imples that $\mathsf{refl} = p$, hence, by the elimination property of R and the fibrancy of $\mathsf{refl} = p$, so does its fibrant replacement.

It therefore follows that for all $p : x = x$, we have $\mathsf{refl} = p$, i.e. A is a set. □

4.4 REEDY FIBRANT DIAGRAMS

In this section we will demonstrate how a two-level system can be used to derive results about HoTT by going outside of the fibrant fragment. This is analogous to how in homotopy theory one can get results that are invariant under homotopy equivalence, even when certain constructions are performed on concrete spaces and do not only depend on their homotopy type.

Specifically, we will define Reedy fibrant diagrams $I \to \mathcal{U}$ for an inverse category I, and show that they have limits in $\mathcal{U}$ if I is finite. This is an internalised version of some of the results in [33].

4.4.1 *Essentially fibrant types and fibrations*

As a preparation for our sample application of the two-level system, we remark that for a strict type $A : \mathcal{U}^{\mathrm{s}}$, asking that A be fibrant is quite a strong requirement. It is often sufficient that there exists a fibrant type $B : \mathcal{U}$ and a strict isomorphism $A \simeq^{\mathrm{s}} B$. If this is the case, we say that A is *essentially fibrant.*

In section 4.2, we have defined the fibrant finite ordinals Fin_n, for $n : \mathbb{N}$, and their strict counterparts $\mathsf{Fin}^{\mathrm{s}}_n$, for $n : \mathbb{N}^{\mathrm{s}}$.

Definition 4.4.1. A type I is said to be *finite* if there exists a number $n : \mathbb{N}^{\mathrm{s}}$ and a strict isomorphisms $I \simeq^{\mathrm{s}} \mathsf{Fin}^{\mathrm{s}}_n$.

Note that Fin_n is not in general finite.

Lemma 4.4.2. *Let I be finite and $X : I \to \mathcal{U}$ be a family of fibrant types. Then, $(i : I) \to X(i)$ is essentially fibrant.*

Proof. Essential finiteness gives us a cardinality n on which we can do induction. If n is 0^{s}, then $(i : I) \to X(i)$ is strictly isomorphic to the unit type. Otherwise, we have an finite I' such that $f : 1 +^s I' \simeq^s I$, and $(i : I) \to X(i)$ is strictly isomorphic to

$$X(f(\mathsf{inl}\ 1)) \times ((i : I') \to X(f(\mathsf{inr}\ i))),$$

which is finite by the induction hypothesis. □

Similar to essential fibrancy, we have the following definition:

Definition 4.4.3. Let $p : E \to B$ be a function. We say that p is a *fibration* if there is a family $F : B \to \mathcal{U}$ such that the fibre of p over any $b : B$ is strictly isomorphic to $F(b)$, that is,

$$(b : B) \to \Big(F(b) \simeq^{\mathrm{s}} (e : E) \times (p(e) \underset{s}{=} b)\Big).$$

Any fibrant type family $F : B \to \mathcal{U}$ gives rise to a fibration $p : E \to B$, as it is easy to see that the first projection $(\Sigma_B F) \to B$ satisfies the given condition. Indeed, any strict fibration is isomorphic over B to a strict fibration of this form. This often allows us to assume that a given fibration has the form of a projection.

4.4.2 *Strict Categories*

We can define categories in a two-level system in much the same way as precategories are defined in [36], except that we can use strict equality to express the laws. Since strict equality does not suffer from coherence issues, this notion of category is well-behaved even when morphisms form a higher type, or even if they are not fibrant at all.

Definition 4.4.4 (strict category)**.** A *strict category* $\mathcal{C}$ is given by:

- a type $|\mathcal{C}|$ of *objects*;
- for all pairs of objects $x, y : |\mathcal{C}|$, a type $\mathcal{C}(x, y)$ of *arrows* or *morphisms*;
- for all objects $x : |\mathcal{C}|$, an *identity* arrow $\mathrm{id} : \mathcal{C}(x, x)$;
- for all objects $x, y, z : |\mathcal{C}|$, a *composition* function

$$\circ : \mathcal{C}(y, z) \to \mathcal{C}(x, y) \to \mathcal{C}(x, z).$$

With the usual categorical laws holding strictly, meaning that we have:

- for all object $x, y : |\mathcal{C}|$ and morphisms $f : \mathcal{C}(x, y)$, strict equalities

$$\begin{aligned} \mathsf{idl} &: f \circ \mathsf{id} \underset{s}{=} f \\ \mathsf{idr} &: \mathsf{id} \circ f \underset{s}{=} f; \end{aligned}$$

- for all objects $x, y, z, w : |\mathcal{C}|$, and morphisms $f : \mathcal{C}(x, y)$, $g : \mathcal{C}(y, z)$, $h : \mathcal{C}(z, w)$, a strict equality

$$\mathsf{assoc} : h \circ (g \circ f) \underset{s}{=} (h \circ g) \circ f.$$

We say that a strict category $\mathcal{C}$ is *locally fibrant* if $\mathcal{C}(x, y)$ is a fibrant type for all objects x, y. We say that $\mathcal{C}$ is *fibrant* if it is locally fibrant and the type of objects is a fibrant type. Finally, we say that $\mathcal{C}$ is *finite* if the type of objects $|\mathcal{C}|$ is finite.

The usual theory of categories can be reproduced in the context of strict categories. It is not hard to define corresponding notions of *functor*, *natural transformation*, *limits*, *adjunctions*, and so on.

From now on, we will refer to strict categories simply as *categories*. If $\mathcal{C}$ is a category, we will often abuse notation and use $\mathcal{C}$ itself to denote its type of objects.

Another important notion is the following:

Definition 4.4.5 (reduced coslice)**.** Given a category $\mathcal{C}$ and an object $x : \mathcal{C}$, the *reduced coslice* $x \mathbin{/\!\!/} \mathcal{C}$ is the full subcategory of non-identity arrows in the coslice category $x/\mathcal{C}$. A concrete definition is the following. The objects of $x \mathbin{/\!\!/} \mathcal{C}$ are triples of the following type:

$$(y : |\mathcal{C}|) \times (f : \mathcal{C}(x, y)) \times \left((p : x \underset{s}{=} y) \to \neg \left(p_*(f) \underset{s}{=} \mathrm{id}\right)\right),$$

where p_* denotes the $\mathsf{transport}$ function $\mathcal{C}(x, y) \to \mathcal{C}(y, y)$, obtained from the eliminator of strict equality. Morphisms between (y, f, s) and (y', f', s') are elements $h : \mathcal{C}(y, y')$ such that $h \circ f \underset{s}{=} f'$ in $\mathcal{C}$.

Note that we have a "forgetful functor" $\mathsf{forget} : x \mathbin{/\!\!/} \mathcal{C} \to \mathcal{C}$, given by the first projection on objects as well as on morphisms.

4.4.3 *Limits and colimits*

Much of what is known about the category of sets in classical category theory can be extended to the category of strict types in a given universe.

For example, the following result translates rather directly:

Lemma 4.4.6. *The universe $\mathcal{U}^{\mathrm{s}}$, regarded as a category in the usual way, has all small limits.*

Proof. Let $\mathcal{C}$ be a category with $|\mathcal{C}| : \mathcal{U}^{\mathrm{s}}$ and $\mathcal{C}(x, y) : \mathcal{U}^{\mathrm{s}}$ (for all x, y), and let $X : \mathcal{C} \to \mathcal{U}^{\mathrm{s}}$ be a functor.

We define L to be the type of natural transformations $1 \to X$, where $1 : \mathcal{C} \to \mathcal{U}^{\mathrm{s}}$ is the constant functor on 1. Clearly, $L : \mathcal{U}^{\mathrm{s}}$, and a routine verification shows that L satisfies the universal property of the limit of X. $\square$

Unfortunately, for colimits the situation is not as pleasant. We can certainly show that $\mathcal{U}^{\mathrm{s}}$ has coproducts, since they can be obtained directly using the strict Σ type structure, but only using our assumptions on the strict fragment of the system, we cannot prove that pushouts exist in $\mathcal{U}^{\mathrm{s}}$.

It would be possible to add pushouts as an additional strict type former. This type former would be set-theoretic, since presheaf models do have arbitrary colimits, so it would not invalidate the assumptions of theorem 3.3.3. Since we will not need arbitrary colimits in the following, we choose to not take this route, and maintain a traditional set of type formers for the strict fragment.

4.4.4 *Inverse Categories*

Classically, *inverse categories* are defined as categories which do not contain an infinite sequence of nonidentity arrows (see [33]).

For simplicity, we restrict ourselves to those which have *height* at most ω, and where a *rank function* is given explicitly. This allows us to perform all constructions constructively, without having to deal with ordinals beyond ω.

First, consider the category $(\mathbb{N}^{\mathrm{s}})^{\mathrm{op}}$ which has $n : \mathbb{N}^{\mathrm{s}}$ as objects, and $(\mathbb{N}^{\mathrm{s}})^{\mathrm{op}}(n, m) :\equiv n >^{\mathrm{s}} m$.

The predicate $>^{\mathrm{s}}: \mathbb{N}^{\mathrm{s}} \to \mathbb{N}^{\mathrm{s}} \to \mathcal{U}^{\mathrm{s}}$ is defined in the familiar way, and it is a *strict proposition*, i.e.

$$(p, q : n >^{\mathrm{s}} m) \to p \underset{s}{=} q$$

Definition 4.4.7. We say that a category $\mathcal{C}$ is an *inverse category* if there is a functor $\varphi : \mathcal{C} \to (\mathbb{N}^{\mathrm{s}})^{\mathrm{op}}$ which "creates identities"; i.e. if we have $f : \mathcal{C}(x, y)$ and $\varphi_x \underset{s}{=} \varphi_y$, then we also have $p : x \underset{s}{=} y$ and $p_*(f) \underset{s}{=} \mathsf{id}$.

4.4.5 *Reedy Fibrant Limits*

We saw in section 4.4.3 that $\mathcal{U}^{\mathrm{s}}$ has all small limits. Unfortunately, the same does not hold for the category $\mathcal{U}$ of fibrant types. Even pullbacks of fibrant types are not fibrant in general (but see Lemma 4.4.8). If we have a functor $X : \mathcal{C} \to \mathcal{U}$, we can always regard it as a functor $X : \mathcal{C} \to \mathcal{U}^{\mathrm{s}}$, where it does have a limit. If this limit happens to be essentially fibrant, we say that X has a *fibrant limit.* Clearly, this limit will then be a limit of the original diagram $C \to \mathcal{U}$, since $\mathcal{U}$ is a full subcategory of $\mathcal{U}^{\mathrm{s}}$.

Of course, the category $\mathcal{U}$ has general *homotopy limits.* For example, given a diagram:

$$\begin{tikzcd} & A \arrow[d, "f"] \\ B \arrow[r, "g"] & C, \end{tikzcd}$$

we can form the corresponding homotopy pullback by taking:

$$P :\equiv (a : A) \times (b : B) \times (f\ a = g\ b),$$

which is fibrant by construction.

It could in principle be possible to use homotopy limits everywhere in place of strict limits, which would therefore work around the question of the existence of strict limits in $\mathcal{U}$. However, defining homotopy limits for general (or even inverse) diagrams already requires some machinery to handle arbitrarily high towers of coherence data, hence we cannot tackle it at this point.

Lemma 4.4.8. *The pullback of a fibration $E \to B$ along any function $f : A \to B$ is a fibration.*

Proof. We can assume that E is of the form $\Sigma\,(b : B)\,.\,C(b)$ and p is the first projection. Clearly, the first projection of $\Sigma\,(a : A)\,.\,C(f(a))$ satisfies the universal property of the pullback. □

Lemma 4.4.8 makes it possible to construct fibrant limits of certain "well-behaved" functors from inverse categories. The so-called *matching objects* play an important role.

Definition 4.4.9 (matching object; see [33, Chp. 11])**.** Let $\mathcal{C}$ be an inverse category, and $X : \mathcal{C} \to \mathcal{U}$ a functor. For any $z : \mathcal{C}$, we define the *matching object* M_z^X to be the (not necessarily fibrant) limit of the composition $z \mathbin{/\!\!/} \mathcal{C} \xrightarrow{\mathsf{forget}} \mathcal{C} \xrightarrow{X} \mathcal{U} \subset \mathcal{U}^{\mathrm{s}}$.

Definition 4.4.10 (Reedy fibrant diagram; see [33, Def. 11.3])**.** Let $\mathcal{C}$ be an inverse category and $X : \mathcal{C} \to \mathcal{U}$ be a functor. We say that X is *Reedy fibrant* if, for all $z : \mathcal{C}$, the canonical map $X_z \to M_z^X$ is a fibration.

Using this definition, we can make precise the claim that we can construct fibrant limits of certain well-behaved diagrams. The following theorem is an internal version of the corresponding result in [33, Lemma 11.8].

Theorem 4.4.11. *Let $\mathcal{C}$ be an finite inverse category. Then, every Reedy fibrant $X : \mathcal{C} \to \mathcal{U}$ has a fibrant limit.*

Proof. By induction on the cardinality of $\mathcal{C}$. If the type of objects is empty, the limit is the unit type.

Otherwise, let us consider the rank functor $\varphi : \mathcal{C} \to (\mathbb{N}^{\mathrm{s}})^{\mathrm{op}}$. We choose an object $z : \mathcal{C}$ such that φ_z is maximal; this is possible (constructively) since $\mathcal{C}$ is assumed to be finite. In particular, z has no incoming arrow (apart from id).

Let us call $\mathcal{C}'$ the category that we get if we remove z from $\mathcal{C}$; that is, we set

$$|\mathcal{C}'| :\equiv (x : |\mathcal{C}|) \times (\neg(x \underset{s}{=} z)).$$

Clearly, $\mathcal{C}'$ is still finite and inverse. Let $X : \mathcal{C} \to \mathcal{U}$ be Reedy fibrant. We can write down the limit of X (i.e. the type of natural transformations to the constant functor) explicitly as

$$(c : (y : |\mathcal{C}|) \to X_y) \times ((y, y' : |\mathcal{C}|)(f : \mathcal{C}(y, y')) \to c_y[f] \underset{s}{=} c_{y'}). \qquad (26)$$

Using that $|\mathcal{C}| \simeq^{\mathrm{s}} |\mathcal{C}'| +^{\mathrm{s}} 1$, and the fact that z has no incoming non-identity arrows, this type is strictly isomorphic to

$$\begin{aligned} &(c_z : X_z) \times (c : (y : |\mathcal{C}'|) \to X_y) \times \\ &\Big((y : |\mathcal{C}'|)(f : \mathcal{C}(z, y)) \to c_z[f] \underset{s}{=} c_y\Big) \times \\ &\Big((y, y' : |\mathcal{C}'|)(f : \mathcal{C}(y, y')) \to c_y[f] \underset{s}{=} c_{y'}\Big). \end{aligned} \tag{27}$$

Let us write L for the limit of X restricted to $\mathcal{C}'$, p for the canonical map $p : L \to M_z^X$, and q for the map $X_z \to M_z^X$.

Then, (27) is strictly isomorphic to

$$(c_z : X_z) \times (d : L) \times (p(d) \underset{s}{=} q(c_z)) \tag{28}$$

This is the pullback of the cospan

$$\begin{tikzcd} L \arrow[r, "p"] & M_z^X & X_z. \arrow[l, "q"'] \end{tikzcd}$$

By Reedy fibrancy of X, the map q is a fibration. Thus, by Lemma 4.4.8, the map from (28) to L is a fibration.

By the induction hypothesis, L is essentially fibrant. This implies that (28) is essentially fibrant, as it is the domain of a fibration whose codomain is essentially fibrant. □

If $\mathcal{C}$ is an inverse category, we will denote by $\mathcal{C}^{<n}$ the full subcategory of $\mathcal{C}$ consisting of all those objects of rank less than n. Correspondingly, for a given diagram X over $\mathcal{C}$, we will denote by $X|n$ the restriction of X to $\mathcal{C}^{<n}$.

4.4.6 *Fibrant Limits and Semi-Simplicial Types*

If X is a Reedy fibrant diagram over $\mathcal{C} :\equiv (\Delta_+^{\mathrm{op}})^{<n}$, we can restrict X to $n \mathbin{/\!\!/} \mathcal{C}$, then take the limit of the corresponding functor. With a slight abuse of notation, we will denote such limit by M_n^X, even though X is not defined at n.

Note that a diagram X over $(\Delta_+^{\mathrm{op}})^{<n+1}$ is Reedy fibrant if and only if its restriction to $(\Delta_+^{\mathrm{op}})^{<n}$ is Reedy fibrant and the map $X_n \to M_n^X$ is a fibration. Hence, to give a Reedy fibrant diagram over $(\Delta_+^{\mathrm{op}})^{<n+1}$ is the same as to give a Reedy fibrant diagram X over $(\Delta_+^{\mathrm{op}})^{<n}$, together with a fibration Y over M_n^X. We will refer to this extended diagram as $\langle X, Y \rangle$.

By mutual induction on the natural number n, we can define a type SST_n, and a function SSK_n from SST_n to diagrams over $(\Delta_+^{\mathrm{op}})^{<n}$. We start with with $\mathsf{SST}_0 :\equiv 1$ and $\mathsf{SSK}_0(1)$ set to the trivial diagram over $(\Delta_+^{\mathrm{op}})^{<0}$.

Then, we set

$$\begin{aligned}\mathsf{SST}_{n+1} &:\equiv \Sigma\,(X : \mathsf{SST}_n)\,.\,(M_n^{\mathsf{SSK}_n X} \to \mathcal{U})\\ \mathsf{SSK}_{n+1}(X,Y) &:\equiv \langle \mathsf{SSK}_n(X), Y\rangle.\end{aligned}$$

Above, we write M_n^A to mean the fibrant type, given by Theorem 4.4.11, which is strictly isomorphic to the matching object of A at n (which would otherwise only be a strict type).

For any strict natural number $n : \mathbb{N}^{\mathrm{s}}$, elements of SST_n are Reedy fibrant n-semi-simplicial types. Since SST_n is fibrant, this gives an internal representation of semi-simplicial types in HoTT.

Unfortunately, unless we add some form of ω-limits to the fibrant fragment of our system, we cannot use the family SST to obtain a fibrant type of general semi-simplicial types (i.e. with simplices of arbitrarily high dimension).

4.5 REEDY-FIBRANT REPLACEMENT

The goal of the current section is to show that any strict functor X from an *admissible* inverse category $\mathcal{C}$ to $\mathcal{U}$ has a fibrant replacement; that is, we can construct a Reedy fibrant diagram which is equivalent in a suitable sense. This construction is an internalisation of the known analogous construction in traditional mathematics (see e.g. [33, Lemma 11.10] or [32].

Note that this notion of fibrant replacement does not contradict the impossibility result of section 4.3. In fact, all the types involved in the construction of a Reedy fibrant replacement are already fibrant: the replacement only happens at the level of diagrams.

Lemma 4.5.1. *Let $f : A \to B$ be a function between fibrant types. Then there exists a fibrant type N, an equivalence $i : A \to N$, and a fibration $p : N \to B$, such that $f \underset{s}{=} p \circ i$.*

Proof. Let $N :\equiv (a : A) \times (b : B) \times (fa = b)$. The function i is given by $i(a) :\equiv (a, f(a), \mathsf{refl})$, while p is simply the projection into the component of type B.

The function i is clearly the inverse of the projection into the component of type A, hence i is an equivalence. Furthermore, p is a fibration, being a projection from a Σ type.

The equation $f \underset{s}{=} p \circ i$ holds definitionally. □

We will refer to the type N constructed in the proof of lemma 4.5.1 as the *mapping cocylinder* of f.

Definition 4.5.2. Let $\mathcal{C}$ be an inverse category. We say that $\mathcal{C}$ is *admissible* if, for all $n : \mathcal{C}$, Reedy fibrant diagrams over the reduced coslice $x \mathbin{/\!\!/} \mathcal{C}$ have a fibrant limit.

The main example of an admissible inverse category is Δ_+^{op}. This follows from Theorem 4.4.11 and the fact that all the reduced coslices of Δ_+^{op} are finite.

Definition 4.5.3. Let X, Y be diagrams over a category $\mathcal{C}$. A natural transformation $f : X \to Y$ is said to be an *equivalence* if, for all $n : \mathcal{C}$, the function $f_n : X_n \to Y_n$ is an equivalence.

Theorem 4.5.4. *Let X be a diagram over an admissible inverse category $\mathcal{C}$. Then there exists a Reedy fibrant diagram Y, and an equivalence $\eta : X \to Y$.*

Proof. We will construct, by induction on the natural number n, a Reedy fibrant diagram $Y^{(n)}$ over $\mathcal{C}^{<n}$, and an equivalence $\eta^{(n)} : X|n \to Y^{(n)}$.

For $n = 0$ there is nothing to construct, so assume the existence of $Y^{(n)}$, and fix any object $x : \mathcal{C}$ of rank $n+1$. The forgetful functor $i_x : x \mathbin{/\!\!/} \mathcal{C} \to \mathcal{C}$ factors through $\mathcal{C}^{<n}$, hence we can consider the composition $Y^{(n)} \circ i$, which is again a Reedy fibrant diagram, and take its limit L.

The map $\eta^{(n)}$ induces a map $X_x \to L$. Define $Y_x^{(n+1)}$ to be the mapping cocylinder of this map. For any object y of rank n or less, define $Y_y^{(n+1)}$ as $Y_y^{(n)}$, and for any morphism $f : \mathcal{C}(x, y)$, the corresponding function $Y_x^{(n+1)} \to Y_y^{(n+1)}$ is given by the projection from the mapping cocylinder, followed by a map of the universal cone of the limit L. The action of $Y^{(n)}$ on morphisms between objects of ranks n or less is defined to be the same as that of $Y^{(n)}$.

It is easy to see that those definitions make $Y^{(n+1)}$ into a diagram that extends $Y^{(n)}$ to objects of rank $n+1$. We can also extend $\eta^{(n)}$ by defining $\eta^{(n+1)}(x)$ to be the embedding of X_x into the mapping cocylinder $Y_x^{(n+1)}$, which is an equivalence by lemma 4.5.1.

Reedy-fibrancy of $Y^{(n+1)}$ follows immediately from the construction, since L is exactly the matching object of $Y^{(n+1)}$ at x.

To conclude the proof, we glue together all the $Y^{(n)}$ and $\eta^{(n)}$ into a single diagram Y and natural transformation η. Clearly, Y is Reedy fibrant, and η is an equivalence. □

4.6 SEMI-SEGAL TYPES

One of the most promising applications of a homotopy type theory with strict equality is the possibility of constructing and working with algebraic objects comprising infinite towers of coherence conditions.

Semi-semplicial types, introduced in section 4.4.6, represent the most fundamental of those objects, and a basis on which to build more complex and directly useful structures.

In this section, we will define the notion of *semi-Segal type* and use it to model $(\infty, 1)$-semicategories internally in HoTT$^{\mathrm{s}}$. The following definitions and results are mostly based on the theory of *Segal spaces* [31], which can, to a certain extent, be thought of as the special case obtained when the model we are working on happens to be the simplicial model (section 3.1).

The caveat here is that, as noted in section 4.4.3, the category of simplicial sets is much richer, in terms of *strict* categorical structure, than what we get to see when working from within type theory. In particular, we noted that the lack of colimits in the formulation of HoTT$^{\mathrm{s}}$ that we adopted makes it really hard (and perhaps impossible) to reproduce the theory of diagrams over general Reedy categories.

Therefore, we cannot hope for a well-behaved theory of Segal types, and we instead settle for the weaker notion of semi-Segal type, which means that we cannot directly model higher categories equipped with identity morphisms, but only semi-category-like structures.

Fortunately, a rich theory can be developed nonetheless. For example, the notion of *completeness*, which superficially seems to require the presence of degeneracies in the underlying simplicial type, can actually be defined for semi-Segal types (definition 4.6.8).

4.6.1 *Preliminaries*

We begin with some definitions concerning semi-simplicial types. Note that a map $f : \Delta^+(n, m)$ is uniquely determined by the finite strictly increasing sequence $f(0), f(1), \ldots, f(n)$. In the following, we will use the notation $\sigma_{f(0),f(1),\ldots,f(n)}$ to denote the face map $X_m \to X_n$ of a semi-simplicial type X corresponding to the map f.

For example, $\sigma_{012} : X_3 \to X_2$ is the face map corresponding to the inclusion $[2] \to [3]$.

For all n, the $n + 1$ face maps $X_n \to X_0$ will therefore be denoted by σ_i, for $i : \mathsf{Fin}_{n+1}$. Finally, for all $i : \mathsf{Fin}_n$, let us write line_i for $\sigma_{i,i+1}$.

Definition 4.6.1. Let X be a semi-simplicial type. The *n-th spine* of X is the type:

$$S_n(X) :\equiv (x : \mathsf{Fin}_n \to X_1) \times ((i : (\mathsf{Fin}_{n-1})) \to \sigma_1(x_i) = \sigma_0(x_{i+1})).$$

The n-th spine of X can be regarded as the type of “paths” of length n in the graph underlying X. Note that $S_n(X)$ is a fibrant type, for all semi-simplicial types X.

Lemma 4.6.2. *Let X be a semi-simplicial type. For all $n : \mathbb{N}$, the family of face maps $\mathsf{line}_i : X_n \to X_1$ determines a map $\phi_n : X_n \to S_n(X)$.*

Proof. It follows from the definition of line_i that:

$$\sigma_1 \circ \mathsf{line}_i \underset{s}{=} \sigma_{i+1} \underset{s}{=} \sigma_0 \circ \mathsf{line}_{i+1},$$

therefore ϕ_n can be defined simply as:

$$\phi_n(x) :\equiv (\lambda i.\mathsf{line}_i(x), \lambda i.\mathsf{refl}).$$

□

The map ϕ_n defined above is called the n-th Segal map of X.

Definition 4.6.3. A *semi-Segal type* is a semi-simplicial type X such that all the Segal maps ϕ_n are equivalences. A morphism of semi-Segal types is simply a morphism of the underlying semi-simplicial types.

For any n, the type expressing the fact that ϕ_n is an equivalence is called the *n-th Segal condition*, and it is a fibrant, propositional type. In fact, being an equivalence is always a proposition ([36]).

For a semi-simplicial type X, the structure of a semi-Segal type is therefore a fibrant proposition, so in particular it is invariant under levelwise equivalence of semi-simplicial types.

We will say that a semi-Segal type is *Reedy fibrant* if the underlying semi-simplicial type is. Note that that the Segal maps of a Reedy fibrant semi-simplicial type are fibrations. The following proposition shows that it is quite easy to obtain Reedy fibrant semi-Segal types.

Proposition 4.6.4. *Let X be a semi-Segal type. Then the Reedy fibrant replacement of X (theorem 4.5.4) is a Reedy fibrant semi-Segal type.*

Proof. Let Y be the Reedy fibrant replacement of X. Since $\eta : X \to Y$ is a levelwise equivalence, we get commutative squares:

$$\begin{tikzcd}
X_n \arrow[r] \arrow[d] & Y_n \arrow[d] \\
X_1 \times_{X_0} \cdots \times_{X_0} X_1 \arrow[r] & Y_1 \times_{Y_0} \cdots \times_{Y_0} Y_1,
\end{tikzcd}$$

where the horizontal maps are equivalences induced by η, and the vertical maps are the Segal maps. Since the Segal maps of X are equivalences, it follows that those of Y are equivalences as well. □

The relationship between semi-Segal types and $(\infty, 1)$-categorical structures on types becomes clear when we analyse the first few levels of their semi-simplicial structure.

Let X be a semi-Segal type. We can think of the type X_0 as the type of *objects* of X. Since X is Reedy fibrant, we have a fibration $\overline{X}_1$ over $M_0(X) = X_0 \times X_0$. The type $\overline{X}_1(x, y)$ can be thought of as the type of *morphisms* between two objects x and y.

So far, we have only singled out a graph. The algebraic nature of semi-Segal types arises from the invertibility of the Segal maps. For all n, let $\psi_n : S_n(X) \to X_n$ be an inverse of ϕ_n. Given morphisms $f : \overline{X}_1(x, y)$ and $g : \overline{X}_1(y, z)$, we can define their *composition* $g \circ f :\equiv \sigma_{02}(\psi(f, g))$, where (f, g) denotes the element of $S_2(X)$ determined by f and g.

With some work, this composition operation can be shown to be *weakly associative*, i.e. there exists a family of *associators*, witnessing equalities between $h \circ (g \circ f)$ and $(h \circ g) \circ f$, for all triples of composable morphisms f, g and h.

In fact, consider the homotopy pullback:

$$\begin{tikzcd}
X_2 \times_{X_1} X_2 \arrow[r, "\pi_1"] \arrow[d, "\pi_2"'] & X_2 \arrow[d, "\sigma_{02}"] \\
X_2 \arrow[r, "\sigma_{01}"'] & X_1.
\end{tikzcd}$$

Using the equivalence ϕ_2 twice, we can easily construct an equivalence $\tau : X_2 \times_{X_1} X_2 \to S_3(X)$. If $u : X_2 \times_{X_1} X_2$ is such that $\tau(u) = (f, g, h)$, then it is not hard to check that $\sigma_{02}(\pi_2(u)) = h \circ (g \circ f)$.

Note that the functions $\sigma_{012}, \sigma_{123} : X_3 \to X_2$ determine a well-defined map $p : X_3 \to X_2 \times_{X_1} X_2$, and $\tau \circ p = \phi_3$. It follows from the 2-out-of-3 property of equivalences that p is also an equivalence.

Now, let $t = \psi_3(f, g, h)$. We have that $(f, g, h) = \phi_3(t) = \tau(p(t))$, hence $p(t) = u$. Therefore, $h \circ (g \circ f) = \sigma_{02}(\pi_2(u)) = \sigma_{03}(t)$. Using a different pullback, one can show that, similarly, $(h \circ g) \circ f = \sigma_{03}(t)$, which implies the required equality.

It is perhaps not surprising that similar arguments, using the Segal conditions at successively higher levels, show the existence of coherence conditions for the semi-categorical structures built so far. For example, at level 4 one can obtain a family of *pentagonators*, witnessing the commutativity of the following diagram of equalities, for all quadruples of composable morphisms f, g, h, k:

$$\begin{tikzcd}
 & k \circ (h \circ (g \circ f)) \arrow[dl] \arrow[dr] & \\
k \circ ((h \circ g) \circ f) \arrow[d] & & (k \circ h) \circ (g \circ f) \arrow[d] \\
(k \circ (h \circ g) \circ f) \arrow[rr] & & ((k \circ h) \circ g) \circ f.
\end{tikzcd}$$

4.6.2 *Nerve of a strict category*

The most fundamental examples of semi-Segal types are given by strict categories. In principle, only *semi-categories* are required, since the identities do not play any role in the construction of the corresponding semi-Segal type. However, we will not be concerned with the extra generality.

Let us recall that in section 4.4.2 we defined a locally fibrant category as a strict category $\mathcal{C}$, such that for all objects x, y of $\mathcal{C}$, the type $\mathcal{C}(x, y)$ is fibrant.

Lemma 4.6.5. *A locally fibrant category $\mathcal{C}$ determines a Reedy fibrant semi-Segal type.*

Proof. We define a semi-simplicial type X using a familiar *nerve* construction:

$$X_n :\equiv (x : \mathsf{Fin}^{\mathrm{s}}_{n+1} \to \mathcal{C}) \times ((i : \mathsf{Fin}^{\mathrm{s}}_n \to \mathcal{C}(x_i, x_{i+1})).$$

Face maps are defined in the usual way. First, given two indices $i, j : \mathsf{Fin}^{\mathrm{s}}_{n+1}$, with $p : i < j$, an pair $(x, f) : X_n$ determines a morphism $f_p : \mathcal{C}(x_i, x_j)$ obtained by composing all the f_k with $i \le k < j$. The composed morphism f_p can easily be defined by induction over the inequality p. Note that X_n is fibrant thanks to lemma 4.4.2.

Now, let $i^+ : i < i + 1$. A map $\sigma : \Delta_+(n, m)$ can be used to obtain an inequality $\sigma(i^+) : \sigma(i) < \sigma(i + 1)$. We can then define $\sigma^* : X_m \to X_n$ as follows:

$$\sigma^*(x, f) :\equiv (\lambda i.x_{\sigma(i)}, \lambda i.f_{\sigma(i^+)}).$$

It is easy to show that X, as defined above, is indeed a semi-simplicial type.

The Segal condition can be shown by directly constructing the equivalence between n-spines and n-simplices. The type of n-spines of X is:

$$(f : \mathsf{Fin}^{\mathrm{s}}_n \to X_1) \times ((i : \mathsf{Fin}^{\mathrm{s}}_{n-1} \to \sigma_1(f_i) = \sigma_0(f_{i+1}))).$$

Expanding the definitions, we get the equivalent type:

$$\begin{aligned}
&((x^0, x^1 : \mathsf{Fin}^{\mathrm{s}}_n \to \mathcal{C}) \\
&\times (f : (i : \mathsf{Fin}^{\mathrm{s}}_n) \to \mathcal{C}(x^0_i, x^1_i)) \\
&\times ((i : \mathsf{Fin}^{\mathrm{s}}_{n-1}) \to x^1_i = x^0_{i+1}).
\end{aligned}$$

We now split x^0 into the pair of $s = x^0_0$ and the rest of the sequence, and similarly split x^1 into the pair consisting of the beginning of the sequence and $t = x^1_{n-1}$. With some index manipulation, this yields the equivalent type:

$$\begin{aligned}
&(s, t : \mathcal{C}) \\
\times&((x^0 x^1 : \mathsf{Fin}^{\mathrm{s}}_{n-1} \to \mathcal{C}) \\
\times&(g :\to \mathcal{C}(s, x^0_0))) \\
\times&(h :\to \mathcal{C}(x^1_{n-2}, t))) \\
\times&(f : (i : \mathsf{Fin}^{\mathrm{s}}_{n-2}) \to \mathcal{C}(x^0_i, x^1_{i+1})) \\
\times&((i : \mathsf{Fin}^{\mathrm{s}}_{n-1}) \to x^1_i = x^0_i).
\end{aligned}$$

The last component of the previous type states that x^0 and x^1 are equal. Therefore, we can contract them into a single sequence x:

$$\begin{aligned}
&(s, t : \mathcal{C}) \\
\times&((x : \mathsf{Fin}^{\mathrm{s}}_{n-1} \to \mathcal{C}) \\
\times&(g :\to \mathcal{C}(s, x_0))) \\
\times&(h :\to \mathcal{C}(x_{n-2}, t))) \\
\times&(f : (i : \mathsf{Fin}^{\mathrm{s}}_{n-2}) \to \mathcal{C}(x_i, x_{i+1})).
\end{aligned}$$

Now we can join s at the beginning of x and t at the end, to get exactly the type X_n as defined above. Examining the equivalence $X_n \to S_n(X)$ obtained by chaining the above steps reveals that it is exactly given by the Segal map, thereby proving that X is a semi-Segal type. $\square$

We call the the semi-Segal type X obtained from a locally fibrant category $\mathcal{C}$ using lemma 4.6.5 the *pre-nerve* of $\mathcal{C}$, and we call *nerve* its Reedy fibrant replacement.

It is important to note that the “weak” categorical structure arising from the nerve X of a locally fibrant category $\mathcal{C}$ matches precisely with the categorical structure on $\mathcal{C}$ itself.

Clearly, the objects and morphisms of X are the same as those of $\mathcal{C}$. Let us now consider composition. Let $f : \mathcal{C}(x_0, x_1)$ and $g : \mathcal{C}(x_1, x_2)$ be two composable morphisms. Their composition as morphisms of the semi-Segal type X is given by applying the face map σ_{02} to the 2-simplex corresponding to the pair (f, g) through the Segal equivalence. It follows from the definition of the semi-simplicial structure on X that this is indeed the composition $g \circ f$, as expected.

Lemma 4.6.5 can be applied to a universe regarded as a strict category. We will denote the nerve of a univalent universe as TYPE, leaving implicit the specific universe used, as usual.

4.6.3 *Maps of semi-Segal types*

The definition of semi-Segal types as semi-simplicial types satisfying a (propositional) property makes it extremely easy to define the corresponding notion of morphism.

Definition 4.6.6. A *semi-Segal map* is a morphism between the underlying semi-simplicial types of two semi-Segal types.

A semi-Segal map can be regarded as the appropriate generalisation of the notion of *functor* between categories. In particular, we can regard a semi-Segal map between the nerves of two strict categories as a *weak semi-functor* between them.

It is important to note that the notion of semi-Segal map between arbitrary semi-Segal types is not fibrant, hence not invariant under equivalence. For example, a map between the *pre-nerves* of two strict categories is the same thing as an ordinary (strict) functor between them, while a semi-Segal map between the nerves is a much weaker notion.

4.6.4 *Completeness*

In the classical theory of Segal spaces, completeness can be understood as the property that the internal notion of *equivalence* in a Segal space can be recovered by only looking the path spaces of its space of points.

In HoTT, completeness is also a very natural property, corresponding to an internal form of univalence for a categorical structure. In [2], completeness is considered such a fundamental property that the term *category* is reserved for those structures that possess it (while those that do not are referred to as *precategories*).

However, it is clear that, in order to define completeness in the setting of semi-Segal types, we first need to derive a notion of *equivalence*, which might appear to be problematic, since semi-Segal types have no identity morphisms.

Fortunately, there is a way to work around this issue:

Definition 4.6.7. Let X be a Reedy fibrant semi-Segal type, and $f : \overline{X}_1(x, y)$ be a morphism. We say that f is an *equivalence* if, for all objects $z : X_0$, the maps:

$$\begin{aligned} f \circ - : \overline{X}_1(z, x) \to X_1(z, y) \\ - \circ f : \overline{X}_1(y, z) \to X_1(x, z), \end{aligned}$$

given by left and right composition with f respectively, are equivalences of types.

It is easy to see that, if X is the nerve of a strict category $\mathcal{C}$, then f is an equivalence if and only if it is a "homotopy equivalence" in $\mathcal{C}$, i.e. if there exists a morphism g in $\mathcal{C}$ in the opposite direction such that $g \circ f = \mathrm{id}$ and $f \circ g = \mathrm{id}$. Note that we are using fibrant equality here, so a homotopy equivalence is not the same as a categorical isomorphism.

The property of being an equivalence for a morphism $f : X_1$ is a mere proposition, denoted $\mathsf{isEquiv}(f)$, hence it determines a *subtype* of X_1:

$$\mathsf{Equiv}^X :\equiv (f : X_1) \times \mathsf{isEquiv}(f).$$

Definition 4.6.8. A Reedy fibrant semi-Segal type X is said to be *complete* if the function

$$\mathsf{Equiv}^X \to X_0,$$

that maps an equivalence to its first endpoint, is an equivalence of types.

Proposition 4.6.9. TYPE *is a complete semi-Segal type.*

Proof. A function $f : X \to Y$ is an equivalence in TYPE if and only if it is an equivalence of types. Therefore, completeness of TYPE follows immediately from univalence. □

A semi-Segal type does not have a built-in notion of identity, but nevertheless, certain morphisms can behave as identities:

Definition 4.6.10. Let $u : \overline{X}_1(x, x)$ be a morphism in a Reedy fibrant semi-Segal type X. We say that u is a *unit* if for all $g : \overline{X}_1(y, x)$ we have that $u \circ g = g$, and for all $h : \overline{X}_1(x, z)$ we have that $h \circ u = h$.

Interestingly, completeness is enough for a semi-Segal type to possess units.

Proposition 4.6.11 (see [15, Lemma 1.4.5])**.** *Let X be a complete semi-Segal type. Then for all objects $x : X_0$ there exists a unit $u : \overline{X}_1(x, x)$.*

Proof. By completeness, we get an object $y : X_0$, and an equivalence $f : \overline{X}_1(x, y)$. Since f is an equivalence, we can find $u : \overline{X}_1(x, x)$ such that $f \circ u = f$. We will show that u is a unit.

If $g : \overline{X}_1(z, x)$, then $f \circ u \circ g = f \circ g$. Using the fact that f is an equivalence again, we get that $u \circ g = g$.

Finally, let $h : \overline{X}_1(x, z)$. We can find $h' : \overline{X}_1(y, z)$ such that $h' \circ f = h$. Then $h \circ u = h' \circ f \circ u = h' \circ f = h$, as required. □

4.7 FURTHER WORK

We have only scratched the surface of what is possible to achieve in a two-level system.

In particular, the notion of semi-Segal type appears to be a quite promising candidate for the role of $(\infty, 1)$-categories in type theory. This thesis only presented the very basic definitions and result, but there is much left to be developed in this area.

BIBLIOGRAPHY

[1] Benedikt Ahrens. Modules over relative monads for syntax and semantics. *ArXiv e-prints*, July 2011.

[2] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science (MSCS)*, pages 1–30, Jan 2015.

[3] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. Categorical structures in type theory, in type theory. Talk at Workshop on Homotopy Theory and Univalent Foundations, Fields Institute, Toronto, Canada, 2016.

[4] Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Extending Homotopy Type Theory with Strict Equality. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62, pages 21:1–21:17, Dagstuhl, Germany, 2016.

[5] S. Awodey. Natural models of homotopy type theory. *ArXiv e-prints*, June 2014.

[6] Andrej Bauer, Gaëtan Gilbert, Philipp Haselwarter, Matija Pretnar, and Chris Stone. Andromeda. Implementation of a type theory with equality reflection.

[7] Marc Bezem, Thierry Coquand, and Simon Huber. A model of type theory in cubical sets, 2014.

[8] Simon Boulier and Nicolas Tabareau. Model structures on types in type theory. in preparation.

[9] Guillaume Brunerie. On the homotopy groups of spheres in homotopy type theory. *ArXiv e-prints*, June 2016.

[10] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209 – 243, 1986.

[11] Peter Dybjer. Internal type theory. In *Types for Proofs and Programs*, pages 120–134. Springer, 1995.

[12] Solomon Feferman. Typical ambiguity: trying to have your cake and eat it too. *One Hundred Years of Russell's Paradox, Berlin: de Gruyter*, pages 135–151, 2004.

[13] Nicola Gambino and Richard Garner. The identity type weak factorisation system. *Theoret. Comput. Sci.*, 409(1):94–109, 2008.

[14] Nicola Gambino and Christian Sattler. The Frobenius Condition, Right Properness, and Uniform Fibrations. *ArXiv e-prints*, October 2015.

[15] Yonatan Harpaz. Quasi-unital ∞–categories. *Algebraic & Geometric Topology*, 15(4):2303–2381, 2015.

[16] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM (JACM)*, 40(1):143–184, 1993.

[17] Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science (MSCS)*, pages 1–16, Mar 2015.

[18] Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and logics of computation (Cambridge, 1995)*, volume 14 of *Publ. Newton Inst.*, pages 79–130. Cambridge Univ. Press, Cambridge, 1997.

[19] Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes.

[20] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *In Venice Festschrift*, pages 83–111. Oxford University Press, 1996.

[21] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after voevodsky). *ArXiv e-prints*, Nov 2012.

[22] Nicolai Kraus. The general universal property of the propositional truncation. In *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, volume 39, pages 111–145, 2015.

[23] Nicolai Kraus and Christian Sattler. Higher homotopies in a hierarchy of univalent universes. *ACM Transactions on Computational Logic (TOCL)*, 16(2):18, 2015.

[24] Peter LeFanu Lumsdaine. *Higher Categories from Type Theories.* PhD thesis, Carnegie Mellon University, 2010.

[25] Jacob Lurie. *Higher Topos Theory*, volume 170 of *Annals of Mathematics Studies.* Princeton University Press, Princeton, NJ, 2009.

[26] Maria Emilia Maietti and Giovanni Sambin. Toward a minimalist foundation for constructive mathematics. *From Sets and Types to Topology and Analysis: Practicable Foundations for Constructive Mathematics*, 48:91–114, 2005.

[27] Per Martin-Löf. An intuitionistic theory of types. In *Twenty-five years of constructive type theory (Venice, 1995)*, pages 127–172. Oxford University Press, 1998.

[28] Ulf Norell. *Towards a practical programming language based on dependent type theory.* PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology and Göteborg University, Göteborg, Sweden, 2007.

[29] Erik Palmgren. Categories with families, FOLDS and logic enriched type theory. *ArXiv e-prints*, May 2016.

[30] Daniel G. Quillen. *Homotopical Algebra*, volume 43 of *Lecture Notes in Mathematics*. Springer-Verlag, 1967.

[31] Charles Rezk. A model for the homotopy theory of homotopy theory. *Trans. Amer. Math. Soc.*, 353(3):973–1007 (electronic), 2001.

[32] Emily Riehl and Dominic Verity. The theory and practice of reedy categories. *Theory and Applications of Categories*, 29(9):256–301, 2014.

[33] Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, pages 1–75, Jan 2015.

[34] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results.* Birkhauser Boston Inc., Cambridge, MA, USA, 1991.

[35] William W. Tait. Intensional interpretations of functionals of finite type I. *The Journal of Symbolic Logic*, 32(2):198–212, 1967.

[36] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[37] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

[38] Vladimir Voevodsky. A simple type system with two identity types, 2013. Unpublished note.

[39] Vladimir Voevodsky. B-systems. *ArXiv e-prints*, October 2014.

[40] Vladimir Voevodsky. Lawvere theories and Jf-relative monads. *ArXiv e-prints*, January 2016.

[41] E. Zermelo. Untersuchungen über die grundlagen der mengenlehre. i. *Mathematische Annalen*, 65:261–281, 1908.