UNIVERSITAT DE BARCELONA

ADVANCED MATHEMATICS
MASTER'S FINAL PROJECT

ACCESSIBLE HIGHER MODALITIES

Author:
Jan Ot Piña Urgell

Supervisor:
Carles Casacuberta

Facultat de Matemàtiques i Informàtica

September, 2023

Abstract

The main goal of this master's thesis is to study localizations in the setting of homotopy type theory. In particular the special case of modalities, an object resembling the namesake notion in classical logic that allows to qualify statements to express things as possibility or necessity.

In the first chapter we introduce the syntax of homotopy type theory and some motivating examples for the study of modalities. Next, we study localizations and modalities as described in [1; 2]. We focus on the fact that accessible modalities correspond to nullifications and work on the existence of non-accessible modalities.

Finally we explore the use of Agda (a programming language for automatic proof verification) towards implementing results about factorization systems and modalities.

Resum

L'objectiu d'aquest treball de final de màster és estudiar les localitzacions en el context de la teoria homotòpica de tipus. En particular el cas de les modalitats, objectes semblants a la noció homònima en lògica clàssica, que permet modificar expressions per a expressar característiques com la possibilitat o la necessitat.

En la primera part de la memòria introduïm la sintaxi de la teoria homotòpica de tipus i alguns exemples motivadors per a l'estudi de les modalitats. Després estudiem les localitzacions i les modalitats tal com es descriuen a [1; 2]. Ens centrem en el fet que les modalitats accessibles corresponen a les nul·lificacions i en l'existència de les modalitats no accessibles.

Finalment explorem l'ús d'Agda (un llenguatge de programació per a la verificació automàtica de demostracions) per a la implementació dels resultats sobre sistemes de factorització i modalitats.

Agraïments

En primer lloc, he d'agrair a en Carles Casacuberta no només la paciència, també tot el que ha aportat al treball, els seminaris i propostes constants mentre érem a Barcelona i la llibertat donada per a girar, rebuscar i canviar l'enfocament d'acord amb els meus interessos i aprenentatge.

També donar gràcies a en Javier amb qui he tingut el plaer de compartir tots els meus avenços i dubtes del primer dia a l'últim minut. Sense les seves aportacions i les seves idees, no només aquesta memòria no hauria sigut possible sinó que la meva manera de veure la teoria de tipus i tot el que l'envolta seria menys ample i molt menys profunda.

Finalment, agrair als professors, companys i amics que m'han acompanyat durant el temps que he estat cursant el màster i que han fet d'aquest curs una vivència tan interessant.

Contents

Chapter 1

Introduction

In general, we refer to localization as the procedure of adding inverses to some algebraic structure, being the classical example the localization of a commutative ring at a set of its elements, making those invertible. The extension of the concept to Category Theory has a long history, starting with the work of Sullivan and Bousfield [3; 4] in Topology and classical Homotopy Theory during the decade of the 1970s and playing a key role in the field.

It is then interesting to present a framework for the study of localization theories in the language of ∞-categories as developed recently by Lurie [5]. In his book *Higher Topos Theory*, Lurie devotes a chapter to the study of the theory of adjoint functors, which, if restricted to the case when one of the functors is an inclusion of a (full) subcategory, it is also the theory of localizations of ∞-categories, and as we will see it is very closely related to the study of *factorization systems*.

Independently of mathematics, classical logic has extensively studied modal logic as a collection of formal systems that aim to represent statements qualifying ways in which they may be considered to be true. As a sample, we might find the classic Alethic Modalities, regarding necessity and possibility. These interact in different ways but the most relevant to us are the two implications :

$$P \to \Diamond P$$

$$\Diamond\Diamond P \to \Diamond P$$

Here we are expressing $\Diamond$ (the possibility modality) as an unary operation on the propositions, namely that *P implies possibly P* and *possibly possibly P implies possibly P*. In category theory or even computer science, we may identify this construction as a *monad*. Many modalities can also take the form of comonads, but we will restrict our attention to the monadic ones.

In the first decade of the 20th century, Bertrand Russell proposed type theory as a solution to certain paradoxes that emerged from naive set theory. Needless to say, set theory would get improved with new formalizations and would get on to be the *standard* for mathematics since then. However type theory would still get developed and worked on; in 1972 Per Martin Löf introduced Intuitionistic Type Theory as an alternative foundation for mathematics based on the principles of constructivism, which revived in many ways the interest and uses of such a formalism.

A type is a concept very similar to a set, as it represents something containing elements. Still, it distances itself from the idea of a set by not allowing the elements (or terms) of a type to be shared with other types or even for these elements to have sense without giving their type. So any element will have one and only one associated type.

Another important difference with respect to set theory is the way to integrate logic. While in set theory we build on first order logic, type theory plays at the same time the role of the sets and the role of the associated logic, via the interpretations of certain types as propositions.

In the late 90s and the first decade of the 21st century, there were many contributions on what the models of such type theories could be. Hofmann and Streicher [6] showed that intuitionistic type theory admitted a model in the category of groupoids. Awodey and Warren showed that Quillen categories also modelled these sorts of type theories, and from this, the homotopy interpretation was developed into a type theory that emerged around a decade ago from the work of Awodey and Warren [7] from one side and Voevodsky [8] from another side. In this homotopic interpretation of type theory, we think of types as spaces or ∞-groupoids. Their terms (or elements) are points in those spaces and for any two points in a space there is a space of paths from one to the other. This type theory makes also use of Voevodsky's *univalence axiom* and *higher inductive types*, which we will explore.

The theory of localizations can be developed with some interesting results in Homotopy Type Theory via higher inductive types and, when restricted to propositions, the notion of a Σ-closed reflective subuniverse in homotopy type theory will coincide with that of a localization in an ∞-topos and the modalities as studied in classical logic.

One of the most relevant modalities in Homotopy Type Theory arises naturally from the *n-truncation* of types, and it corresponds to the localization at the function from $\mathbb{S}^{n+1}$ (the $n+1$-sphere) to the unit type. More generally given a family of functions $F : B(a) \to C(a)$ indexed by the terms of a type $a : A$, a type X is F-local if the map between the types of functions

$$(C(a) \to X) \to (B(a) \to X)$$

induced by precomposition is an equivalence for all $a : A$ indexing the family F. An F-localization of a type X is a universal F-local type admitting a map from X, we will say that a modality is *accessible* if it arises from an F-localization

When localizing at a family of maps a reflective subuniverse will arise, but it will not necessarily be a modality. We can though impose a condition to make sure that we obtain a modality, namely that the maps are constant. We call these localizations *nullifications*, following the usual homotopical terminology, with n-truncation as an example. Every accessible modality can be presented as a nullification.

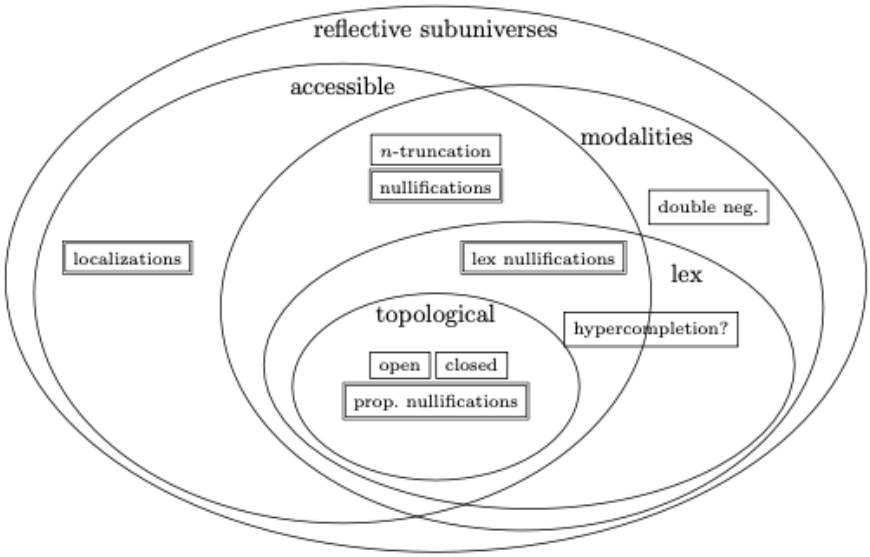


Figure 1.1: Modalities and related structures, from [2]

The non-accessible localizations are less known. In [2] the example of the double negation modality is discussed:

$$P \to \neg\neg P.$$

Despite being accessible under the correct assumptions on the *size* of types, it does not seem to always be. Another interesting result is that, again under size assumptions, we can construct localizations whose accessibility is independent of ZFC, a construction that we can extend to modalities.

The final chapter is dedicated to the formalization of Homotopy Type Theory, as it opens up a new direction on the capabilities of foundations of mathematics and computation, allowing high dimensional data types and an effective implementation into a programming language.

Its applications go from allowing us to work with the "right" notion of equality for mathematical structures to expressing higher mathematical concepts as higher types The computational proof assistants developed on this basis facilitate the large-scale formalization of logic and mathematics into libraries, with great practical implications for mathematics.

Maybe the best examples are the Hott/Coq-HoTT library, based on the Coq proof assistant, with around 60 contributors to it, and the Agda-Unimath library, based on the Agda language and with 32 contributors. These two libraries are not as active or big as other projects (see for example mathlib and the Xena project) but they formalize work done from a univalent point of view, that is, using Homotopy Type Theory.

Chapter 2

Fundamentals of Homotopy Type Theory

The objective of this chapter is to introduce the vocabulary and how-to's of *homotopy type theory* to someone who has not yet worked on it or attended a course. We will focus on the *syntax* of the theory, which is its formal presentation as a logical system, rather than its interpretation in a given model.

We will follow mainly the formalism of the book "Homotopy Type Theory: Univalent Foundations of Mathematics" [9]. This book gives a great introduction to Homotopy Type Theory from a "mathematical use" of it, in the same way we usually make use of set theory. Other options for the reader to delve into Homotopy Type Theory are the recent "Introduction to Homotopy Theory" [10] and the recorded lectures of the HoTTEST Summer School 2022

Intuitionistic Type Theory (or Dependent Type Theory) is a sort of type theory introduced in the 70s by Per Martin Löf [11], which makes use of dependent types, constructive logic and identity types. This kind of type theory together with the modelling of types as ∞-groupoids and the introduction of Voevodsky's univalence axiom and Higher Inductive Types is what is given the name of Homotopy Type Theory.

2.1 Dependent Type Theory

Type theory is a deductive system and, as such, we have judgements and rules, that allow us to introduce and construct new types and terms from what we already have. There is only a rather small set of basic judgements:

A is a type	A : Type
a is a term of type A	a: A
A and B are equal types	A≡ B: Type
a and b are equal terms of type A	a≡ b: A
P(a) is a type for every a:A	a:A $\vdash$ P(a) :Type
p(a): P(a) for every a:A	a:A $\vdash$ p(a) :P(a)

Remark 1. "*Type*" is a universe, a type that has other types as terms. To avoid certain paradoxes existing also in the set theoretic realm, it is usual to suppose a hierarchy of universes

$$U_0 : U_1 : U_2 : U_3 \dots$$

In practice, we only use the symbol $\mathcal{U}$ to denote a certain universe unless other universes are needed. Therefore from now on we will write $A : \mathcal{U}$ or simply A for "let A be a type".

Note that we are considering two sorts of judgements, those of introduction of a term given its type and those of *definitional* equality, that is, given a type A, and two terms $x, y : A$, it is valid to say $x = y : A$ which we can also write for example $y :\equiv x$ when we are giving an expression. This is equality in its strongest form, but not the one we will use where we would use equality in the sense of set theory or "usual mathematics". We will see later propositional equality, which we will denote $\cdot =_A \cdot$ and is internal to the type theory. There are types $Id_A(x, y)$ and when an element of this type can be constructed we say that x and y are *propositionally* equal.

A Martin-Löf dependent type theory makes use of propositional equality but also introduces the two last judgements of the list. The types (and terms) introduced can be thought of as analogous of indexed sets, which in type theory we will call *dependent types*. We can think of p in the second judgement as a function taking an element $a : A$ to $p(a) : P(a)$. We call such function a dependent function.

Here the part on the left of the turnstile $\vdash$ is the "context", the given premises for our judgement. We will not make use of that language but an extended explanation for both context and judgements can be found in [10].

Concerning the other part of the "system", *rules* characterize how a given type and its terms behave. We will see plenty of examples since before using type theory we need to define basic types and constructions. We will first give some types we consider part of the fundamentals of type theory which we will make use of during the rest of this text. We will then jump to the homotopy interpretation of type theory and its implications and axioms.

Function types

In set theory, we define a function as a set of pairs. This sort of definition allows us to deduce certain properties that characterize functions and how we will work with them. In type theory they are a *primitive* concept, so we will explain the type by giving the rules, defining functions from what is expected from them rather than the other way around.

Definition 1. *Let $A, B : \mathcal{U}$ be two types. We define the type of* functions with domain A and codomain B, *denoted as $A \to B$, as the one following the rules:*

- ***Introduction**: given a free variable and an expression θ which may contain x, such that for any element $a : A$ then $\theta[a/x] : B$ then the equation*

  $$f(x) :\equiv \theta$$

  is a term $f : A \to B$.

- ***Elimination**: If $a : A$ and $f : A \to B$ then we can apply the function f to the element a, and we get $f(a) : B$, which we call the image of a.*

- ***Computation**: Given $f(x) :\equiv \theta$ a term of $A \to B$, for any $a : A$ we have*

  $$f(a) \equiv \theta[a/x] : B.$$

Example 1. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by the equation $f(x) :\equiv x + x$. Then $f(2)$ is judgmentally equal to $2 + 2$.

A notation that we will use a lot and that allows us to not give names to functions is that of λ-*abstraction*, given an expression θ of type B we write $\lambda.(x : A).\theta$ to refer to the same function given in the introduction rule. Following the example above we would have

$$\lambda(n : \mathbb{N}).x + x : \mathbb{N} \to \mathbb{N}$$

We will usually omit the type from the variable and give it implicitly.

2.1.1 Dependent type over a type

Let $A : \mathcal{U}$, as we have already stated. a function taking each of the terms $a : A$ to a type $P(x) : \mathcal{U}$ is said to be a dependent type over A. These types are present in mathematics and we will make abundant use of them, for example as a way of taking a *family* of types with some property. From dependent types we can also form distinct and useful types:

Dependent function types

Let $P : A \to \mathcal{U}$ be a dependent type over A. We can form the *dependent function* type $\Pi_{(a:A)}P(a)$ also called Π-type. The terms of this type are functions whose codomain type varies depending on the element applied to it. If P is a constant family, we recover the ordinary functions type:

$$\Pi_{(a:A)}P \equiv (A \to P)$$

Being more formal, we can give the rules which determine the behaviour of this type:

Definition 2. *Let A be a type and $P : A \to \mathcal{U}$ a dependent type over A. We define the* type of dependent function with domain A and codomain P, *denoted $\Pi_{(a:A)}P(a)$, as the type obeying the following rules:*

- ***Introduction**: We can introduce a dependent function explicitly, given a free variable x and an expression θ that may contain x, if for every $a : A$, $\theta[a/x] : P(a)$ then*

  $$f(x) :\equiv \theta$$

  introduces a dependent function $f : \Pi_{(x:A)}P(x)$.

- ***Elimination**: As with usual functions, we can apply an argument $a : A$ to $f : \Pi_{(x:A)}P(x)$, and obtain an element $f(a) : B(a)$. We will call $f(a)$ the image of f for a.*

- ***Computation**: Given a function $f : \Pi_{(x:A)}P(x)$ defined as $f(x) :\equiv \theta$, for all $a : A$, we will have*

  $$f(a) \equiv \theta[a/x] : B(a).$$

Dependent pair types

In the same way that we generalized functions to dependent functions, we can generalize the product, allowing the second component to vary along the first component. Dependent pair types or Σ-types would correspond in set theory to an indexed sum over a given type. Given $A : \mathcal{U}$ and $P : A \to \mathcal{U}$ we will write $\Sigma_{(a:A)}P(a)$. As before with functions, if we take a constant family we recover the usual product:

$$\Sigma_{(a:A)}P \equiv A \times B.$$

Definition 3. *Let A be a type and $P : A \to \mathcal{U}$ a dependent type over A. We define the* type of dependent pairs, *denoted $\Sigma_{(a:A)}P(a)$ as the type obeying the following rules:*

- ***Introduction**: We construct elements of the type by pairing, that is, given $a : A$ and $p : P(a)$ we have $(a, p) : \Sigma_{(a:A)}P(a)$.*

- ***Elimination**: Given a pair (a,p), we have elements $pr_1((a,p)) : A$ and $pr_2((a,p)) : P(a)$. Note that*

$$pr_1 : \Sigma_{(a:A)} P(a) \to A$$

and as the image of the projection on the second element depends on the element a, it is a dependent function:

$$pr_2 : \Pi_{(w:\Sigma_{(a:A)} P(a))} P(pr_1(a,p)).$$

- ***Computation and Uniqueness**: Given $a : A$ and $p : P(a)$,*

$$pr_1(a,p) = a : A$$

$$pr_2(a,p) = p : P(a)$$

and given a pair (a,p), then $(a,p) = (pr_1(a,p), pr_2(a,p))$.

Remark 2. In both this definition and the previous one we have not introduced all the rules properly. For example, the explicit formation rules that explain what we need to have a well-formed type, but only what we need to work with the language both in concept and from a practical perspective. We adopt this point of view in this thesis for brevity but also to not get lost in type theory, since in most of the work we focus on it from a mathematical point of view rather than the formalism we are using.

2.1.2 Other types

Coproduct type

One of the types we will make use of is the *coproduct type*, which would be in the place of the disjoint union in set theory or the coproduct in category theory. Given any two types $A, B : \mathcal{U}$, we define $A + B$ as their coproduct.

For this type we get two introduction rules. The first says that given $a : A$ we can introduce a term $inl(a) : A + B$. In the same way, the second says that given $b : B$ we can introduce a term $inr(b) : A + B$ (inr and inl are short for left injection and right injection).

We can form also functions of type $A + B \to C$, given functions $f_1 : A \to C$ and $f_2 : B \to C$, let:

$$\begin{aligned} f(inl(a)) &:\equiv f_1(a) \\ f(inr(b)) &:\equiv f_2(b) \end{aligned}$$

So we are defining the function by cases, in the same way we can construct a family a dependent function of type $\Pi_{x:A+B} C(x)$ by taking $g_1 : \Pi_{x:A} C(x)$ and $g_2 : \Pi_{x:B} C(x)$:

$$\begin{aligned} g(inl(a)) &:\equiv g_1(a) \\ g(inr(b)) &:\equiv g_2(b). \end{aligned}$$

Empty type

The empty type, which we denote $\varnothing$ or $\mathbf{0}$, is one that has many similitudes with initial objects in category theory and obviously given the notation, with the empty set. Since it has no elements we give no introduction rule (or any). We can nevertheless construct a function $f : \mathbf{0} \to C$ for any C, without an equation defining it since the type has no elements from which to define an image. In the same way we can define a dependent function $f : \Pi_{x:0} C(x)$.

This is one of the types we will use under the *propositions as types* paradigm. For example, the dependent function previously defined corresponds to the logical principle *ex falso quodlibet*. Moreover, we use this type to define negation:

Definition 4. *For any type $A : \mathcal{U}$ we define its negation $\neg A$ as*

$$\neg A :\equiv A \to \varnothing.$$

Since $\neg A$ is the type of functions from A to $\varnothing$, a proof of $\neg A$ is given by assuming that A holds, and then constructing an element of the empty type. So we prove it by deriving a contradiction. This proof technique is called proof of negation and it is different from a proof by contradiction, where we would show that $\neg A$ yields a contradiction, involving the step that $\neg\neg A$ implies A. Note though that $\neg\neg A$ is the same as a function of type

$$(A \to \varnothing) \to \varnothing.$$

So we cannot assert without more information that this is the same as A, and therefore we might not be able to construct $\neg\neg A \to A$. Therefore we are dealing with a deductive system where the *law of excluded middle* does not hold.

Unit type

The unit type $\mathbf{1}$ is the type with only one element, which we denote $* : \mathbf{1}$. Following with the analogies to category theory, it would make the role of a final object. This type has only one introduction rule, namely the one that allows the introduction of its only element, and given any $c : C$ we can define a function by $f(*) :\equiv c$. Then we can also construct

$$f : \Pi_{x:\mathbf{1}} C(x)$$

also given by the assignation $f(*) :\equiv c$.

Most of those last types we have seen are instances of inductive types, which we will discuss in section 2.3. Before that, we will see the last component of a Martin Löf dependent type theory.

2.1.3 Propositions as types

As we mentioned in the Introduction, showing that a proposition is true in type theory is the same as the construction of an element of the type that corresponds to this proposition. We regard the elements of this type as a *witness* or *proof*. We can interpret any type as a proposition but this will only make sense in part of them, as we will see.

The observation that gives rise to the types as propositions principle is the following correspondence between logical operations on propositions and *type theoretic* operations.

Natural Language	Type Theory
True	$\mathbf{1}$
False	$\mathbf{0}$
Not A	$A \to \mathbf{0}$
A and B	$A \times B$
A or B	$A + B$
If A then B	$A \to B$
A iff B	$(A \to B) \times (B \to A)$

The point of the table above is that the rules for formation, and use of the terms of the type on the right, correspond to the rules for reasoning with the proposition on the left, and from there the *witness of proof* name. For instance given a witness of a function type $f : A \to B$ and a witness $a : A$, we have "proof of A implies B" and "proof of A" from what we can deduce B. We can also make use of the *elimination rule* of function types, and apply f to the witness $a : A$, producing $f(a) : B$, a witness of B.

We have in the table instances of propositional logic, if we want to add quantifiers, *for all* or *there exists*. In this case, types are used both as propositions and also the domain we quantify over, as usual types. A predicate over a type A is represented by a dependent family $P : A \to \mathcal{U}$ that assigns to every element $a : A$ a type $P(x)$ corresponding to the proposition that P holds for a. Then we can extend the prior table:

Natural Language	Type Theory
for all $x : A$, $P(x)$ holds	$\Pi_{(x:A)}P(x)$
there exists $x : A$ such that $P(x)$ holds	$\Sigma_{(x:A)}P(x)$

The constructivity of type-theoretic logic means that we have some freedom to introduce axioms to our context. For example, while by default we have no LEM in constructive logic, it is still consistent with the existence of one (as we will see later). We might add it to our assumptions and work with it. A formulation of the classical law of excluded middle* in a universe would be the following:

$$LEM_\infty :\equiv \Pi_{(A:\mathcal{U})} A + \neg A.$$

We will make use of that when introducing other axioms such as the univalence axiom.

Moreover, we have described here a *proof relevant* logic which takes proofs as objects themselves carrying information about the proof. For instance, in a witness of $A \times B$, "A or B" we would know whether the witness is a term of A or a term of B, or similarly in a proof of "there exists $x : A$ such that $P(x)$" written as a term $(x, p) : \Sigma_{(x:A)}P(x)$ we know which element x is by taking the projection.

As a consequence of this logic we might have constructions where the proposition exhibit very different behaviours. For example taking "$\mathbb{N} \iff \mathbf{1}$", in type theoretical language $(\mathbb{N} \to \mathbf{1}) \times (\mathbf{1} \to \mathbb{N})$. This type is inhabited as it is easy to construct examples of both function types. What this statement tells us is that when *treated as propositions* those two types are equivalent. We often refer to this as *logically equivalent*, to distinguish it from the stronger notion of a type equivalence. In the following chapter we will introduce a modification of this system by introducing *mere propositions*, a class of types where logical equivalence and type equivalences are the same.

2.1.4 Equality: identity types

The use of types to interpret the logical features would suggest that there is also a natural way to code the interpretations of equality into type theory. This interpretation comes as identity types, introduced first by Per Martin Löf. According to our propositions as types paradigm, the proposition that two elements $a, b : A$ are equal has to correspond to a dependent family $Id_A(-,-) : A \to A \to \mathcal{U}$ dependent on two copies of A since it can vary for both elements. Then we might write $Id_A(a, b)$ as the type representing the proposition that a equals b and we will say that *a and b are propositionally equal*. For clarity we will write $a =_A b$ since the equality is given in an "ambient" type, but we can write it without the type if there is no risk of confusion.

Over the sets we can define equality to be the least reflexive relation on that set. We can see that the idea is similar when dealing with types, since we only have an introduction rule. For all types A and element $a : A$ there is an element

$$refl_a : a =_A a.$$

The elimination rule for identity types states (in a reduced form) that we can substitute equals:

Indiscernibility of equals: For every family

$$P : A \to \mathcal{U}$$

*This formulation of the law of excluded middle is actually inconsistent with HoTT, for it would render all the types trivial. See the link.

there is a function

$$f : \Pi_{(x,y:A)}\Pi_{(p:x=_A y)}P(x) \to P(y).$$

such that

$$f(x, x, refl_x) = Id_{P(x)}.$$

However in order to fully characterize the behaviour of the identity type we have to consider not only maps out of $x = y$ but also families over the type. The *path induction* principle, named that way referring to the homotopy interpretation, can be seen as the property of identity types being freely generated by $refl_x$.

Path induction: For every family

$$P : \Pi_{x,y:A}(x =_A y) \to \mathcal{U}$$

and function

$$p : \Pi_{(x:A)}P(x, x, refl_x)$$

there is a function

$$f : \Pi_{(x,y:A)}\Pi_{(q:x=_A y)}C(x, y, q)$$

such that $f(x, x, refl_x) :\equiv c(x)$.

Note that if we have a definitional equality $x \equiv y$, it follows from path induction that x and y are propositionally equal.

There is something that does not quite add in this definition of the identity types. Since $a =_A b$ is a type given two elements $a, b : A$, there can be more than one identification $a = b$. This raises the question of how we think of equalities when we have multiple identifications between two elements. One helpful way to think about it is to imagine that types are spaces, and their elements are points in those spaces. An identification is then a path in the space, connecting the two end points. This idea lands into a concretion in the homotopy interpretation of homotopy type theory:

2.2 Homotopy Type Theory

As we have already stated, homotopy type theory is an extension of intuitionistic type theory, with the principal difference being the treatment of types as spaces in the abstract homotopy setting, more specifically as ∞-groupoids, the higher categorical vesrion of groupoids.

In topology a path in a topological space is a continuous map $p : [0, 1] \to X$, taking $p(0) = x$ as the starting point and $p(1) = y$ as its ending point. Since the point-wise identification of paths is in many cases not useful for it is too fine, we take the relation given by homotopies, the continuous maps $\alpha : [0, 1] \to ([0, 1] \to X)$ such that $\alpha(0) = p_0$ and $\alpha(1) = p_1$ where p_i are paths between the same starting and ending points. Here α is a path between paths, a homotopy. Since a homotopy has also a path-like structure, we can take a homotopy between homotopies, and go on to 3-homotopies, and define inductively n-homotopies for any n.

The base space X, the paths, and the n-homotopies for all n form an ∞-groupoid, called the fundamental ∞-groupoid of X. An ∞-groupoid is an ∞-category (with higher morphisms) where all the morphisms are invertible. The theory of ∞-categories, although very useful when discussing semantics of homotopy type theory and modalities, is way out of scope for this thesis so we will only use the fundamental ∞-groupoid as our working example.

Let $A : \mathcal{U}$, and recall that for any $x, y : A$ we can take the identity type $x =_A y$. Logically we can think about terms of $x =_A y$ as witnesses of the equality, but type theory allows us to go further and treat these proofs as objects allowing them to be subjects of other propositions such as $p =_{x=_A y} q$. We can iterate this construction further, and the object arising from this tower of equalities corresponds to the fundamental ∞-groupoid on the space A.

2.2.1 Types as spaces

All of the structure of an ∞-groupoid can be deduced from the induction principle on identity types. We will show symmetry of identity, which in an ∞-groupoid would be the same as reversing paths, and transitivity of equality, which corresponds to concatenation.

Lemma 1. For every type A and terms $x, y : A$ there is a function

$$(x =_A y) \to (y =_A x)$$

denoted by $p \mapsto p^{-1}$ and such that $refl_x \equiv refl_x^{-1}$ for each $x : A$.

Proof. We want to get from $x, y : A$ and $p : x = y$ an element $p^{-1} : y = x$. By the induction principle it is sufficient to show it for the case where y is x and p is $refl_x$, but then both identity types are $x = x$ and we can take $refl_x^{-1}$ to be simply $refl_x$. The general case follows by induction. □

Remark 3. Recall that a proposition in type theory corresponds to a type, so the statement of a lemma should be able to be translated into a type and its proof translated into an inhabitant of that type, so the proof in a more formal language would be the obtention of a term of type:

$$\Pi_{A:\mathcal{U}}\Pi_{x,y:A}(x = y) \to (y = x)$$

(the second part $refl_x^{-1} \equiv refl_x$ is a separate proposition, but we can get it from the construction of the witness). Then take a family $C : \Pi_{x,y:A}(x = y) \to \mathcal{U}$ defined by $D(x, y, p) :\equiv (y = x)$ and an element $d :\equiv \lambda x.refl_x : \Pi_A D(x, x, refl_x)$. Then the path induction principle allows us to construct an element

$$ind_=(D, d, x, y, p) : (x = y)$$

for each p, so we take the function to be $\lambda p.ind_=(D, d, x, y, p)$.
We can even translate this into a "wordless" proof, but the use of natural language such as the one used in the first proof is usually preferred.

Lemma 2. Let A be a type and $x, y, z : A$. Then there exists a function

$$(x = y) \to (y = z) \to (x = y)$$

written as $p \to q \mapsto p \cdot q$, such that $refl_x \cdot refl_x \equiv refl_x$ for any $a : A$. We call $p \cdot q$ the *concatenation* or *composite* of p and q.

Proof. We want to construct for every $x, y, z : A$ and $p : x = y$, $q : y = z$ an element of $x = z$. By induction on p it is suficient to assume $x \equiv y$ and $p :\equiv refl_x$, and doing induction again on q, $x \equiv y \equiv z$ and $q \equiv p \equiv refl_x$, in this case we have $refl_x : x = z$, and the general case follows by path induction. □

Then we have the following *correspondences* table:

Equalities	Homotopy	∞-Groupoid
reflexivity	constant path	identity morphism
symmetry	inversion of paths	inverse morphism
transitivity	concatenation	composition morphism

From these three properties, we can also prove the following properties showing that our operations are well behaved. Let $A : \mathcal{U}$, $x, y, z, w : A$ and $p : x = y$, $q : y = z$, $r : z = w$. Then:

i) $p = p \cdot refl_y$ and $p = refl_x \cdot p$

ii) $p^{-1} \cdot p = refl_y$ and $p \cdot p^{-1} = refl_x$

iii) $(p^{-1})^{-1} = p$

iv) $p \cdot (q \cdot r) = (p \cdot q) \cdot r$

All the proofs for these properties follow again from the rules of the identity type.

2.2.2 Functions as functors

Given that we consider types to be ∞-groupoids, it is natural to think that function types should correspond to functors between them and therefore act accordingly. In type theory this is the statement that functions preserve equalities (and n-equalities for every n), that is, given an equality we will have an image of such, being an equality of the images.

Lemma 3. Let $f : A \to B$. For every $x, y : A$ there is a function

$$ap : (x =_A y) \to f(x) =_B f(y).$$

Moreover, for each $x : A$ we have $ap_f(refl_x) = refl_{f(x)}$.

Proof. Again by path induction it is enough to give it for the case where $x \equiv y$ and $p :\equiv refl_x$, and by the identity type introduction rule we know that there is $refl_{f(x)} : f(x) =_B f(x)$, and $ap_f(refl_x) :\equiv refl_{f(x)}$. □

Now we need to check the rest of functoriality:

Lemma 4. Let $f : A \to B$, $g : B \to C$, $p : x =_A y$ and $p : y =_A z$:

i) $ap_f(p \cdot q) = ap_f(p) \cdot ap_f(q)$

ii) $ap_f(p^{-1}) = ap_f(p)^{-1}$

iii) $ap_g(ap_f(p)) = ap_{g \circ f}(p)$

iv) $ap_{id}(p) = p$

Proof. Again all the proofs follow from the path induction principle, so we only have to consider the reflective case and the rest follows from definitions.

i) Here we have to do induction on both paths:

$$ap_f(refl_x \cdot refl_x) \equiv ap_f(refl_x) \equiv refl_{f(x)} \equiv refl_{f(x)} \cdot refl_{f(x)} \equiv ap_f(refl_x) \cdot ap_f(refl_x).$$

ii) By induction with $x \equiv y$ and $refl_x$, we obtain

$$ap_f(refl_x^{-1}) \equiv ap_f(refl_x) \equiv refl_{f(x)} \equiv refl_{f(x)}^{-1} \equiv ap_f(refl_x)^{-1}.$$

iii) By induction with $x \equiv y$ and $refl_x$, we obtain

$$ap_g(ap_f(refl_x)) \equiv ap_g(refl_{f(x)}) \equiv refl_{g(f(x))} \equiv ap_{g \circ f}(refl_x).$$

iv) By induction with $x \equiv y$ and $refl_x$, we obtain

$$ap_{id}(refl_x) \equiv refl_{id(x)} \equiv refl_x.$$

□

Remark 4. If we follow with the comparisons and correspondences between homotopy type theory, we can ask the same question about applying equality to functions when talking about dependent functions. In such a case we would find out that a dependent function f over a family E corresponds to the sections of a fibration defined by the family E in the associated ∞-groupoid.

Recall that the interpretation of types as spaces is not the only addition of Homotopy Type Theory from a more standard dependent type theory. Later in this work we will get to see higher inductive types and before that the *univalence axiom*.

2.2.3 Type equivalences

Until now we have seen how equality between elements induces a structure on types. But we want to see also the situation of equalities and equivalences between types and especially function types. Usually, we consider the pointwise equality $\Pi_{x:A} f(x) = g(x)$ for two $f, g : A \to B$. If the type is inhabited then the two functions should be equal. More generally:

Definition 5. *let $f, g : \Pi_{x:A} P(x)$ be two dependent functions of a type family $P : A \to \mathcal{U}$. A homotopy from f to g is a dependent function type,*

$$(f \simeq g) :\equiv \Pi_{x:A} f(x) = g(x).$$

These newly defined homotopies between functions correspond to the natural transformations in the ∞-groupoid setting.

Note that a pointwise identification (or homotopy) between two functions is not the same as $f =_{\Pi_{x:A} P(x)} g$, but there is the following function.

Lemma 5. Let $A, B : \mathcal{U}$ and two functions $f, g : A \to B$. There exists a function,

$$happly : (f = g) \to (f \sim g)$$

given by the proof.

Proof. We use induction by paths, assuming $f \equiv g$ and $refl_f$. In this case we have $(f \simeq f) \equiv \Pi_{x:A}(f(x) = f(x))$ so we have a mapping $x \mapsto refl_{f(x)} : f \simeq f$. The general case follows by induction. □

Getting back to types, the concept of an isomorphism in the usual sense is having two $f : A \to B$ and $g : B \to A$ such that both composites $g \circ f$ and $f \circ g$ are pointwise the identity, i.e. $g \circ f \sim id_A$ and $f \circ g \sim id_B$. This is the type that would code the concept of a *homotopy equivalence* in type theory or a *equivalence of higher groupoids.*

Definition 6. *For a function $f : A \to B$, a* quasi-inverse *of a f is a triple (f, α, β) consisting of a function $g : B \to A$ and homotopies $\alpha : f \circ g \sim id_B$ and $\beta : g \circ f \sim id_A$. Thus the type of quasi-inverses is the following:*

$$\Sigma_{(g:B\to A)}(f \circ g \sim id_B) \times (g \circ f \sim id_A).$$

But the type of quasi-inverses is not well-behaved, since we can have multiple inhabitants. So we would want another definition of equality $isEquiv$, such that:

- $qinv(f) \to isEquiv(f)$
- $isEquiv(f) \to quinv(f)$
- $isEquiv(f)$ has either none or one element (it is a *mere proposition* as we will see later).

Definition 7. *Given $f : A \to B$, we say that f is an* equivalence *if the following holds:*

$$isequiv(f) : (\Sigma_{(g:A\to B)} f \circ g \sim id_B) \times (\Sigma_{(h:B\to A)} h \circ f \sim id_A).$$

Definition 8. *Given two types A, B, we define the type of equivalences between A and B*

$$(A \simeq B) :\equiv \Sigma_{(f:A\to B)} isequiv(f),$$

Although this is the most direct instance of the equivalence notion we were looking for, we can use other variations, like the following.

Definition 9. *We say that f is* path-split *if we have an inhabitant of the following type:*

$$pathsplit(f) :\equiv rinv(f) \times \Pi_{x,y:A} rinv(ap_f^{x,y})$$

where $rinv(f) :\equiv (\Sigma_{(g:A\to B)} f \circ g \sim id_B)$ and $ap_f^{x,y} : (x = y) \to (f(x) = f(y))$.

Theorem 5. *For any f, $pathsplit(f) \simeq isEquiv(f)$.*

Proof. If f is path-split, we have already the right inverse, so we have to show that its right inverse g is also a left inverse, that is, $g(f(x)) = x$ for all $a : A$, but $fgf(x) = f(x)$ since $f \circ g = id_B$, and $ap_f : (gf(x) = x) \to fgf(x) = f(x)$ has a right inverse so $g(f(x)) = x$. Thus we got a map $pathsplit(f) \to isEquiv(f)$. The other direction follows from properties of contractibility that we will see later. □

We will make use of this definition of equivalence later on.

The Univalence Axiom

Given two types $A, B : \mathcal{U}$ we might consider them as elements of the type $\mathcal{U}$ and therefore form an equality type $A =_{\mathcal{U}} B$. As we have done with homotopy and equalities applied to functions, we want to investigate the relation between equalities and equivalences of types. We begin by showing that there is a function between them:

Lemma 6. Let $A, B : \mathcal{U}$. There is a function

$$idtoeqv : (A =_{\mathcal{U}} B) \to (A \simeq B)$$

given in the proof.

Proof. Note that we can see $id_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}$ as a family assigning every type to itself. Thus given a path $p : A = B$ we have a transport function $p_* : A \to B$. Our claim is that p_* is an equivalence, but then by induction we only have to asssume that $p \equiv refl_A$ in which case the induced function is the identity, and is then an equivalence, so we define $idtoeqv(p)$ as p_* together with the proof that it is an equivalence. □

We would like for *idtoeqv* to be an equivalence, but the type theory we defined does not suffice to guarantee this, so we include it as the *univalence axiom*:

Axiom 1. For any $A, B : \mathcal{U}$, the function in Lemma 6 is an equivalence, that is,

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

.

The axiom refers to the particular universe the types inhabit, so we will talk about $\mathcal{U}$ being an univalent universe when this holds. The implications of this axiom are many and very different but one that we will use in the rest of the work is the following

Theorem 6. *(Function extensionality) Let $A, B : \mathcal{U}$ and $f, g : A \to B$. There is an equivalence:*

$$(f \sim g) \to (f = g).$$

Therefore two functions are equal if they coincide at every point.

2.2.4 Propositions as (some) types

We have seen that types behave as ∞-groupoids. We can find though certain subclasses of types that do not have any meaningful information from a certain level of equalities and upwards.

We obtain these types by taking the equality level to be discrete. In particular we use the following definition, which is related to the classical notion in topology:

Definition 10. *We say that a type $A : \mathcal{U}$ is* contractible *if*

$$isContr(A) :\equiv \Sigma_{a:A} \Pi_{x:A} (x = a),$$

this is, there exists $a : A$ such that for every $x : A$ we have $a = x$. We also call contractibel types (-2)-types. We call a center *of A, the projection $pr_1(A)$ given $A : isContr(A)$.*

Another subclass we can define is the one of *mere propositions*. These types when interpreted as propositions work as the propositions in set theory. Unlike the contractible types which are always inhabited, mere propositions can be inhabited or not, but when they are all their elements are the same one.

Definition 11. *A type A is a mere proposition if for all $x, y : A$ we have that $x = y$, that is,*

$$isProp(A) :\equiv \Pi_{x,y:A}(x = y).$$

We also call mere propositions (-1)-types.

Lemma 7. If P and Q are mere propositions such that $P \to Q$ and $Q \to P$, then $P \simeq Q$.

Proof. Suppose given $f : P \to Q$ and $g : Q \to P$. Then for any $x : P$ we have $g(f(x)) = x$ since P is a mere proposition, and the same goes for $y : Q$ and $f(g(y)) = y$; therefore f and g are quasi-inverses. □

Corollary 1. If P is a mere proposition and it is inhabited (that is we have $x : P$) then $P \simeq \mathbf{1}$.

Proof. Since $\mathbf{1}$ is a mere proposition, we can define for $x : P$ the functions $\lambda * .x : \mathbf{1} \to P$ and $\lambda x.* : P \to \mathbf{1}$. □

The logic generated by the interpretation of propositions as types, does not match fully with the logic we have in classsical set theory, to obtain a similar logic we can restrict us to interpreting as propositions only those types being mere propositions. The logical connectors we have seen work well with this interpretation.

Finally, we can define a type that behaves like a set, passing to the next step of the equality ladder. Instead of discretizing the elements we will do the deed with the equalities of elements, this is: a set will be a type such that an equality between two elements is unique.

Definition 12. *We say that a type A is a set if*

$$isSet(A) :\equiv \Pi_{x,y:A}\Pi_{p,q:x=y}(p = q).$$

We also call sets 0-types.

We will see a generalization of this concept, n-types, in the following chapter.

2.3 Inductive Types

As we have seen in the previous sections, the definition of a new type can require quite an amount of work, since we have to specify all the rules it follows. But in some cases we can *generate* those types from one or more constructors. We will call those types *inductive types*.

To talk about inductive types we first have to define what a constructor is. A *constructor* is a function with codomain an inductive type X. This has into account functions with an empty argument, which give simply elements of X. Note that when evaluated a constructor will always give an element of the inductive type.

Example 2. The type of booleans $\mathbf{2}$ is inductively generated by the constructors

- $0_2 : \mathbf{2}$
- $1_2 : \mathbf{2}$

Another example where the constructor functions take arguments is the coproduct.

Example 3. The type $A + B$ is inductively generated by the constructors

- $inl : A \to A + B$
- $inr : B \to A + B$

And more importantly we can take arguments of the constructors from the same inductive type.

Example 4. The type of natural numbers $\mathbb{N}$ is inductively generated by the constructors

- $0_{\mathbb{N}} : \mathbb{N}$
- $succ : \mathbb{N} \to \mathbb{N}$

Then terms of inductive types are obtained by repeatedly applying the given constructors. In this sense we would expect the terms of $\mathbb{N}$ to be either zero or an element obtained from applying the successor function to a previously known term of $\mathbb{N}$.

Together with the constructors, inductive types are also determined by their *induction principle* and *computation rule*. The induction principle, also called *dependent elimination rule*, specifies which data shoud be provided in order to be able to construct a section of a type family over the inductive type. In order to define a dependent function $f : \Pi_{(x:X)} B(x)$, one has to specify the behaviour of f at all the constructors of X.

So for example if we have a family $B : \mathbf{2} \to \mathcal{U}$ to build an $f : \Pi_{x:\mathbf{2}} E(x)$,, we have to choose two elements $e_0 : E(0_{\mathbf{2}})$ and $e_1 : E(1_{\mathbf{2}})$ such that $f(0_{\mathbf{2}}) = e_0$ and $f(1_{\mathbf{2}}) = e_1$, so if E is a proposition we get the usual proof by cases.

The computation rule asserts that the inductively obtained section agrees on the constructors. So we get one for each of them. Following with the same example, the computation rules are

$$ind_{\mathbf{2}}(E, e_0, e_1, 0_{\mathbf{2}}) \equiv e_0$$

$$ind_{\mathbf{2}}(E, e_0, e_1, 1_{\mathbf{2}}) \equiv e_1.$$

Let us see a case with constructors with an argument.

Example 5. The most obvious example of an inductive type is the natural number type, $\mathbb{N}$, as defined before. We also have two constructors as in $\mathbf{2}$, but the case analysis is not enough. In the case corresponding to the inductive step $succ(n)$, we also want to presume that the statement we want to prove has been shown already for n. We have then the induction principle:

- A statement about natural numbers $P : \mathbb{N} \to \mathcal{U}$ can be proven showing it for $0_\mathbb{N}$ and for "$succ(n)$ given it holds for n", that is, we construct:

$$p : E(0_\mathbb{N})$$
$$p_s : \Pi_{n:\mathbb{N}} E(n) \to E(succ(n))$$

and as we defined them, we have associated computation rules for the function $ind_\mathbb{N}(E, p, p_s) : \Pi_{x:\mathbb{N}} E(x)$,

- $ind_\mathbb{N}(E, p, p_s, 0) \equiv p$
- $ind_\mathbb{N}(E, p, p_s, succ(n)) \equiv p_s(n, ind_\mathbb{N}(E, p, p_s, n)$ (for any $n : \mathbb{N}$).

The dependent function $ind_\mathbb{N}(E, p, p_s)$ can be though as being defined recursively on the argument $x : \mathbb{N}$ using p and p_s. If x is zero then we simply get p, but when it is the successor of another number, we take the recurrence p_s and substitute the predecessor n and the recursive $ind_\mathbb{N}(E, p, p_s, n)$

2.3.1 Higher Inductive Types

A limitation that constructors in inductive types have is that they will only allow us to construct terms of such types. Since we are working in the setting of homotopy type theory, we would want to define a type specifying not only their elements or *points*, but also the *paths* that work as identifications, both for points or with higher instances. This generalization of the inductive types receives the name of a *higher inductive type* and it is one of the pillars of homotopy type theory.

The definition of a higher inductive type will be the same as that of an ordinary inductive type, with the difference that we will allow constructors that generate elements in the types of equalities above our constructed type. Therefore the constructors can have a codomain different from the inductive type.

We will refer to the ordinary sort of constructors (such as the ones we have already used for inductive types) as *point constructors* and to the others as *path constructors* or *higher constructors*.

Example 6. The S^1 type is generated by a point constructor

- $base : S^1$

and a path constructor

- $loop : base =_{S^1} base$.

Propositional truncation

We have already seen what role *mere propositions* play in homotopy type theory. Propositional truncation or *squash* or *bracket type* can be thought as an operation that truncates a type down to being a mere proposition, leaving only the information about the type being inhabited or not. We will introduce it as a first instance of higher inductive types in this work.

Definition 13. *Let A be a type. We define its* propositional truncation $\|A\|$ *to be the higher inductive type generated by,*

- *A function* $\|-\| : A \to \|A\|$,
- *for each* $x, y : \|A\|$ *a path* $x = y$.

Note that the second constructor just means that $\|A\|$ is a mere proposition. The recursion principle for this higher inductive type is the following: given any type B together with

- a function $g : A \to B$,
- for any $x, y : B$ a path $y =_B x$,

there exists a function $f : \|A\| \to B$ such that

- $f(|a|) \equiv g(a)$ for all $a : A$, and
- for any $x, y : \|A\|$, the function $ap_f : x =_A y \to f(x) =_B f(y)$ takes the specified path $x = y$ in $\|A\|$ to the specified path in $f(x) = f(y)$ in B.

The induction principle for propositional truncation says that given a family $P : \|A\| \to \mathcal{U}$ together with

- a function $g : \Pi_{(a:A)} B(|a|)$,
- for any $x, y : \|A\|$ and $p : P(x)$, $p' : P(y)$, a dependent path $q : p =^P_{s(x,y)} p'$ where $s(x, y)$ is the path given by the second constructor of $\|A\|$,

there exists $f : \Pi_{x:\|A\|} P(x)$ such that $f(|a|) \equiv g(a)$ for $a : A$.

Since there is only at most one function between two mere propositions, this induction principle does not have much use, but we can extend this idea of propositional truncation to n-types, as mere propositions are (-1)-types, i.e. types with trivial homotopy above the (-1) level.

2.4 n-truncations

The structure we obtain from the generalizations of propositional truncation and the factorizations systems they induce will be a motivating example of the main object we will study in this thesis, namely higher modalities.

The idea is to make use of the following facts about loop spaces and map spaces:

- A is an n-type if and only if $\Omega^{n+1}(A, a)$, the type of loops based at $a : A$, is contractible for all $a : A$.
- $\Omega^{n+1}(A, a) \simeq Map_*(\mathbb{S}^{n+1}, (A, a))$, the type of loops with a basepoint is equivalent to the one of maps from $\mathbb{S}^{n+1}$ to the pointed type.

These two theorems and more detailed properties of the types given can be found in [9, Section 2.4 and 6.7]. Now we can give the definition of n-truncation:

Definition 14. *For $n \geq -1$ we take $\|A\|_n$ to be the higher inductive type generated by*

- *a function $\|-\|_n : A \to \|A\|_n$*
- *for each $r : \mathbb{S}^{n+1} \to \|A\|_n$ a point $h(r) : \|A\|_n$*
- *for each $r : \mathbb{S}^{n+1} \to \|A\|_n$ and each point $x : \mathbb{S}^{n+1}$ a path $s_r : r(x) = h(r)$.*

Lemma 8. $\|A\|_n$ is an n-type.

Proof. It suffices to show that $\Omega^{n+1}(\|A\|_n, b)$ is contractible for any $b : \|A\|_n$, which is equivalent to $Map_*(\mathbb{S}^{n+1}, (\|A\|_n, b))$. We choose the constant map at b, $c_b : \mathbb{S}^{n+1} \to \|A\|_n$, together wit h $refl_b : c_b(base) = b$. An element of $Map_*(\mathbb{S}^{n+1}, (\|A\|_n, b))$ is a map $r : \mathbb{S}^{n+1} \to \|A\|_n$ with a path $p : r(base) = b$. To show $r = c_b$ it suffices to get for any $x : \mathbb{S}^{n+1}$ a path $r(x) = c_b(x) = b$. We take this to be $s_r(x) \cdot s_r(base)^{-1} \cdot p$. We have to show that when transported along this equality $r = c_b$ the path p becomes $refl_b$. Hence we need

$$(s_r(x) \cdot s_r(base)^{-1} \cdot p) \cdot p^{-1} = refl_b,$$

But this follows directly from path composition. □

We get the inductive principle from the constructors, given a $P : \|A\|_n \to \mathcal{U}$ together with

- for each $a : A$ an element $g(a) : P(|a|_n)$,
- for each $r : \mathbb{S}^{n+1} \to \|A\|$ and $r' : \Pi_{(x:\mathbb{S}^{n+1})} P(r(x))$, an element $h'(r, r') : P(h(r))$,
- for each $r : \mathbb{S}^1 \to \|A\|_n$ and $r' : \Pi_{(x:\mathbb{S}^{n+1})} P(r(x))$, and each $x : \mathbb{S}^{n+1}$, a path $r'(x) =^P_{s_r(x)} h'(r, r')$,

then there is a section $f : \Pi_{x:\|A\|_n} P(x)$ with $f(|a|_n) \equiv g(a)$ for all $a : A$. We can reformulate the induction principle as it follows:

Theorem 7. *Given a $P : \|A\|_n \to \mathcal{U}$ where every $P(x)$ is an n-type, and any function $g : \Pi_{(a:A)} P(|a|_n)$, then there is a section $f : \Pi_{(x:\|A\|_n)} P(x)$ such that $f(|a|_n) \equiv g(a)$ for all $a : A$.*

Proof. Since g is the same as the first bullet point in the data of the inductive principle, we only need to build the other two. Given $r : \mathbb{S}^1 \to \|A\|_n$ and $r' : \Pi_{(x:\mathbb{S}^{n+1})} P(r(x))$, we have $h(r) : \|A\|_n$ and $s_r : \Pi_{(x:\mathbb{S}^{n+1})}(r(x) = h(r))$. Define $t : \mathbb{S}^{n+1} \to P(h(r))$ by $t(x) :\equiv s_r(x) * (r'(x))$, join.Then since $P(h(r))$ is n-truncated there is a point $u : P(h(r))$ and a contraction $v : \Pi_{(x:\mathbb{S}^{n+1})}(t(x) = u)$. We define $h'(r, r') :\equiv u$, getting the second point. Then v has the type of the third point. □

The induction principle also implies a uniqueness principle for functions of this form. If E is an n-type and $g, g' : \|A\|_n \to E$ such that $g(|a|_n) = g'(|a|_n)$ for every $a : A$ then $g(x) = g'(x)$ for all $x : \|A\|_n$ as $\|A\|_n$ is an n-type. Therefore $g = g'$. This gives us the following universal property

Theorem 8 (Universal property of truncations)**.** *Let $n \geq -1$, $A : \mathcal{U}$ and $B : n\text{-Type}$. Then there is a map*

$$\begin{aligned} (\|A\|_n \to B) &\to (A \to B) \\ g &\mapsto g \circ \|-\|_n \end{aligned}$$

Proof. Given that B is n-truncated any $f : A \to B$ can be extended to a map $ext(f) : \|A\|_n \to B$ and the map $ext(f) \circ \|-\|_n$ is equal to f, since for any $a : A$ we have $ext(f)(\|a\|_n) = f(a)$ by definition and the map $ext(g \circ \|-\|_n)$ is equal to g because they both send $\|a\|_n$ to $g(\|a\|_a)$. □

In the setting of category theory we would say that n-types form a *reflective category*, and in particular the n-truncation is functorial. We have for example $\|f \circ g\|_n = \|f\|_n \circ \|g\|_n$ and $\|Id_A\|_n = Id_{\|A\|_n}$, and also higher instances of functoriality on paths. Moreover we have the following property:

Theorem 9. *A type $A : \mathcal{U}$ is an n-type if and only if $\|-\| : A \to \|A\|_n$ is an equivalence.*

Along many more properties that we will see in the next chapter on modalities.

2.5 Orthogonal Systems

In set theory we can factorize any map uniquely as a surjection followed by an injection. We will see a natural generalization of injective and surjective maps in the setting of homotopy type theory, and then the corresponding generalization of the unique factorization system of maps between sets.

2.5.1 n-connectedness

We can see n-types as types that do not have any information above the n^{th} dimension. We can also take an n-connected type as the one that has no information *below* dimension n. It is natural to study this behaviour also for maps. We introduce first the fiber of a map over a point.

Definition 15. *The fiber of a map $f : A \to B$ over a point $y : B$ is*

$$fib_f(y) :\equiv \Sigma_{x:A}(f(x) = y).$$

In homotopy theory we would call this the homotopy fiber *of f.*

Definition 16. *A function $f : A \to B$ is said to be* n-connected *if for all b the type $\|fib_f(b)\|_n$ is contractible, that is,*

$$conn_n(f) :\equiv \Pi_{b:B} isContr(\|fib_f(b)\|)$$

A type is n-connected if the function $A \to \mathbf{1}$ is n-connected, that is if $\|A\|_n$ is contractible.

Therefore a function is n-connected if and only if the fiber at b is n-connected for every $b : B$. Taking the (-1) level we will retrieve surjections, as f is surjective if $\|fib_f(b)\|_n$ is inhabited for all $b : B$, but in a mere proposition inhabitation is the same as contractibility.

Therefore we can think of n-connectedness as a stronger or higher form of surjectivity.

Lemma 9. For any $A : \mathcal{U}$, the function $\|-\|_n : A \to \|A\|_n$ is n-connected.

2.5.2 Orthogonal factorizations

Since we have seen n-connectedness as a generalization of surjection, the corresponding notion for injective functions is the following.

Definition 17. *A function $f : A \to B$ is n-truncated if the fiber fib_f is an n-type for all $b : B$.*

Lemma 10. For any $n \geq -2$ a function $f : A \to B$ is $(n+1)$-truncated if and only if for all $x, y : A$, the map $ap_f : (x = y) \to f(x) = f(y)$ is n-truncated. In particular f is (-1)-truncated if and only if it is an embedding.

Proof. For any $(x, p), (y, q) : fib_f(b)$ we have

$$\begin{aligned}((x,p) = (y,q)) &= \Sigma_{r:x=y}(p = ap_f(r) \cdot q) \\ &= \Sigma_{r:x=y}(ap_f(r) = p \cdot q) \\ &= fib_{ap_f}(p \cdot q^{-1})\end{aligned}$$

Thus, any path space in a fiber of f is a fiber of ap_f. On the other hand by choosing $b :\equiv f(y)$ and $q = refl_{f(y)}$, we see that a fiber of ap_f is a path space in a fiber of f. As f is n-truncated if all path spaces of their fibers are n-types. □

We can now describe the factorization system:

Definition 18. *Let $f : A \to B$ be a function. The n-image of f is defined as*

$$im_n(f) :\equiv \Sigma_{(b:B)} \|fib_f(b)\|_n.$$

When $n = -1$ we simply call it the image of f and write $im(f)$.

Lemma 11. For any function $f : A \to B$ the function $f : A \to im_n(f)$ is n-connected. As a consequence any function factors as a composition of an n-connected and a n-truncated function.

Proof. Note that $A \simeq \Sigma_{b:B} fib_f(b)$. Take the function $\widetilde{f}$ on total spaces, induced by the fiberwise transformation.

$$\Pi_{(b:B)} fib_f(b) \to \|fib_b(f)\|.$$

We have $fib_{\widetilde{f}}((b, v)) \simeq fib_{f(b)}(v)$ for each $b : B$ and $v : \|fib_b(f)\|$, hence $\left\|fib_{\widetilde{f}}((b,v))\right\|_n$ is contractible if and only if $\left\|fib_{f(b)}(v)\right\|$. Since $fib_f(b) \to \|fib_b(f)\|$ is n-connected this implies that $\widetilde{f}$ is n-connected. Finally the projection $pr_1 : im_n(f) \to B$ is n-truncated, since its fibers are equivalent to the n-truncation of the fibers of f. □

It remains to see whether this factorization is unique as in the case of sets. But before we prove a technical result.

Proposition 1. Given a commutative diagram

$$\begin{array}{ccc} A & \xrightarrow{g_1} & X_2 \\ \downarrow{\scriptstyle g_2} & & \downarrow{\scriptstyle h_1} \\ X_2 & \xrightarrow{h_2} & B \end{array}$$

with $H : h_1 \circ g_1 \sim h_2 \circ g_2$ as a witness of commutativity, where the g_i are n-connected and the h_i are n-truncated functions, there is an equivalence:

$$E(H, b) : fib_{h_1}(b) \simeq fib_{h_2}(b)$$

for any $b : B$, such that for any $a : A$ we have an identification

$$\bar{E}(H, b) : E(H, h_1(g_1(a))(g_1(a), refl_{h_1(g_1(a))}) = (g_2(a), H(a)^{-1}).$$

Proof. Let $b : B$, then we have the following equivalences:

$$\begin{aligned} fib_{h_1}(b) &\simeq \Sigma_{(w:fib_{h_1}(b))} \left\| fib_{g_1(pr_1(w))} \right\|_n \\ &\simeq \left\| \Sigma_{(w:fib_{h_1}(b))} fib_{g_1(pr_1(w))} \right\|_n \\ &\simeq \left\| fib_{h_1 \circ g_1}(b) \right\|_n \end{aligned}$$

with the first equivalence coming from g_1 being n-connected, the second from h_1 being n-truncated and finally the last equivalence can be deduced from the properties of fibrations. We have likewise equivalences for g_2 and h_2, and also as we have $H : h_1 \circ g_1 \sim h_2 \circ g_2$ we have an equivalence

$$fib_{h_1 \circ g_1}(b) \simeq fib_{h_2 \circ g_2}(b).$$

Hence we get

$$fib_{h_1}(b) \simeq fib_{h_2}(b)$$

for any $b : B$. Now if we take an element $(g_1(a), refl_{h_1(g_1(a))})$, when taking the image of all the equivalences that composed form E we obtain

$$\begin{aligned} (g_1(a), refl_{h_1(g_1(a))}) &\mapsto ((g_1(a), refl_{h_1(g_1(a))}), \left\| (a, refl_{g_1(a)}) \right\|_n) \\ &\mapsto \left\| ((g_1(a), refl_{h_1(g_1(a))}), (a, refl_{g_1(a)}) \right\|_n \\ &\mapsto \left\| (a, refl_{h_1(g_1(a))}) \right\|_n \\ &\mapsto \left\| (a, H(a)^{-1}) \right\|_n \\ &\mapsto \left\| ((g_2(a), H^{-1}(a)), (a, refl_{g_2(a)})) \right\|_n \\ &\mapsto ((g_2(a), H^{-1}(a)), \left\| (a, refl_{g_2(a)}) \right\|_n) \\ &\mapsto (g_2, H(a)^{-1}). \end{aligned}$$

□

With this proposition we can prove the following uniqueness theorem:

Theorem 10. *For each $f : A \to B$, the space $fact_n(f)$ defined by*

$$\Sigma_{X:\mathcal{U}}\Sigma_{g:A\to X}\Sigma_{h:X\to B}(h \circ g \sim f) \times conn(f) \times trunc_n(f)$$

is contractible. Its center of contraction is the term

$$(im_n(f), \hat{f}, pr_1, \theta, \varphi, \varsigma)$$

that is shown in Lemma 11, with $\theta : pr_1 \circ \hat{f} \sim f$ the canonical homotopy, φ the proof of Lemma 11 and ς the proof of $pr_1 : Im_n(f) \to B$ having truncated fibers.

Proof. By the aforementioned lemma, we know that there is an element of the type $fact_n(f)$. So we only have to show that the type is a mere proposition. Take two factorizations:

$$(X_1, g_1, h_1, H_1, \varphi_1, \vartheta_1)$$

and

$$(X_2, g_2, h_2, H_2, \varphi_2, \vartheta_2)$$

Then we have the concatenated homotopy

$$H :\equiv (\lambda a.H_1(a) \cdot H_2^{-1}(a)) : (h_1 \circ g_1 \sim h_2 \circ g_2.$$

It is sufficient to show the following:

- There is an equivalence $e : X_1 \simeq X_2$.
- There is an homotopy $\xi : e \circ g_1 \sim g_2$.
- There is an homotopy $\eta : h_2 \circ e \sim h_1$.
- For any $a : A$ we have $ap_{h_2}(\xi(a))^{-1} \cdot \eta(g_1(a)) \cdot H_1(a) = H_2(a)$.

We will prove it in this order since each one depends on the previous data.

1. By the previous lemma we have a fiberwise equivalence

   $$E(H) : \Pi_{b:B} fib_{h_1}(b) \simeq fib_{h_2}(b)$$

   that in its turn induces an equivalence of total spaces, i.e.,

   $$(\Sigma_{b:B} fib_{h_1}(b)) \simeq (\Sigma_{b:B} fib_{h_2}(b).$$

   Since we also have equivalences $X_1 \simeq \Sigma_{(b:B)} fib_{h_1}(b)$ and $X_2 \simeq \Sigma_{(b:B)} fib_{h_2}(b)$, we get our e explicitly:

   $$e(x) :\equiv pr_1(E(H, h_1))(x, refl_{h_1(x)}).$$

2. Using the previous lemma again we can choose $\xi(a) :\equiv ap_{pr_1}(\bar{E}(H, a)) : e(g_1(a)) = g_2(a)$.

3. For every $x : X_1$ we have

   $$pr_1(E(H, h_1(x))(x, refl_{h_1(x)})) : h_2(e(x)) = h_1(x).$$

   giving the homotopy η.

4. The path $\bar{E}(H, a)$ from the previous lemma gives us $\eta(g_1(a)) = ap_{h_2}(\xi(a)) \cdot H(a)^{-1}$. By substituting H with its definition and rearranging paths we end the proof.

□

Chapter 3

Localizations and Modalities in Homotopy Type Theory

If we translate to type theory the concept of an *idempotent monad* on a universe we obtain the notion of a *reflective subuniverse*. Some of those reflective subuniverses, as the n-truncation, are also closed under the formation of Σ-types. The notion of a reflective subuniverse is enough to model the concept of *idempotent monad on* $\mathcal{U}$. Regardless of that, the condition of it being Σ-closed is included in the definition of a modality. It is natural to include that condition as it is equivalent to having a *dependent elimination rule* similar to the ones we can find in inductive types and makes the different characterizations of modalities arise.

3.1 Reflective Subuniverses

Definition 19. *A subuniverse is a predicate* $isLocal : \mathcal{U} \to Prop$. *If for a* $X : \mathcal{U}$ *the type* $isLocal_L(X)$ *is inhabited we will say that* X *is* L-local *and we write* $\mathcal{U}_L$ *for the subuniverse of* L-local *types.*

Definition 20. *Given a subuniverse* L *and a type* X, *an* L-localization *of* X *consists of an* L*-local type* X' *and a map* $g : X \to X'$ *such that for every* L*-local type* Y *the precomposition map*

$$g^* : (X' \to Y) \to (X \to Y)$$

Is an equivalence. We will call this the universal property of L-localization.

Lemma 12. Given a subuniverse L, the type of L-localizations of X is a mere proposition.

Proof. Our data consists of $Y : \mathcal{U}_L$, $f : X \to Y$ and $I : \Pi_{(X':\mathcal{U})} IsEquivalence(\lambda g.g \circ f : (X' \to Y) \to (X \to Y))$. To show that this triple is a mere proposition we have to show that if we have another triple (Y', f', I'), then $(Y', f', I') = (Y, f, I)$.

As I, I' are terms of a mere proposition, if we show $(Y, f) = (Y', f')$ we have this third equality for free. To show the latter all we need is to find $g : Y \to Y'$ such that $g \circ f' = f$, with g an equivalence.

By I restricted to Y', we have that the type of pairs (g, h) where $g : Y \to Y'$ and such that $h : g \circ f = f'$ is a contractible type. The same is true for the type of pairs (g', h') where $g : Y' \to Y$ and $h : g' \circ f' = f$.

Now $g' \circ g$ is such that $g' \circ g \circ f = g' \circ f' = f$, and it follows by contractibility that $g' \circ g = Id_Y$. With the same argument we get $g \circ g' = Id_{Y'}$, therefore g is an equivalence. □

Definition 21. *A* reflective subuniverse L *consists of a subuniverse* $isLocal_L : \mathcal{U} \to \mathbf{Prop}$, *a reflector* $L : \mathcal{U} \to \mathcal{U}_L$ *and a* unit

$$\eta : \Pi_{(X:\mathcal{U})} X \to LX$$

such that for every X *the map* $\eta_X : X \to LX$ *is an* L*-localization of* X.

Remark 11. The comparable object to a reflective subuniverse in category theory would be that of a *reflective subcategory*, a full subcategory $\mathcal{C}'$ of $\mathcal{C}$ such that the inclusion i has a left adjoint L called the reflector. This is what we also call a *localization*.

Corollary 2 (Corollary to Lemma 12). Two reflective subuniverses with the same local types are the same.

Proof. If we have the same local types, then for each of those types $X : \mathcal{U}$, we have a triple (X, η_X, I) such as the one in Lemma 12. Therefore by the lemma this triple forms a mere proposition. □

Corollary 3. Given a reflective subuniverse, a type $X : \mathcal{U}$ is local if and only if η_X is an equivalence.

Proof. If the unit is an equivalence then X is local as it is equivalent to a local type. Conversely if X is local we have $(X, id_X, -)$ inhabiting the type from Lemma 12 which is also inhabited by $(LX, \eta_X, -)$, but the type is a mere proposition, so both terms are equal and η_X is an equivalence. □

3.1.1 Functorial and *nice* properties

We would want our localizations as in the traditional sense to behave in a functor-like manner. We will also show that any reflective subuniverse is closed under pullbacks and identity types, contains the unit type, and dependent products.

Lemma 13. Given $f : A \to B$ we have an induced $Lf : LA \to LB$ that preserves identity and composition up to homotopy. Moreover for any f the square

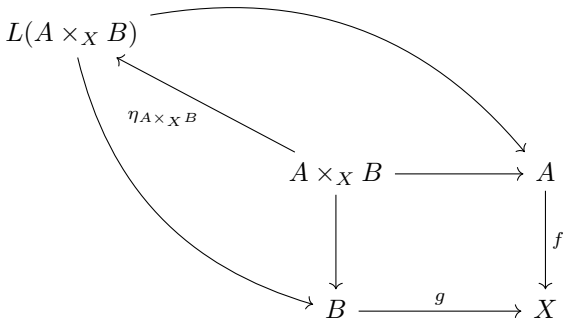
$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ LA & \xrightarrow{Lf} & LB \end{array}$$

commutes.

Proof. We take Lf to be the unique function such that $Lf \circ \eta_A = \eta_B \circ f$ given by the universal property of localization for η_A. We can use the same argument for the preservation of the identity and composition. □

Lemma 14. Any reflective subuniverse is closed under pullback.

Proof. Let X be a modal type and take two maps $f : A \to X$ and $g : B \to X$. In the diagram



the outer square commutes as both morphisms extend $A \times_X B \to X$ along the unit, and by the universal property of localizations this extension is unique. Therefore by the universal property of the pullback, we get a f left inverse to $\eta_{A \times_X B}$. Then $(\eta_{A \times_X B} \circ f) \circ \eta_{A \times_X B} = \eta_{A \times_X B}$ so $\eta_{A \times_X B} \circ f$ is a factorization of $\eta_{A \times_X B}$ through the same $\eta_{A \times_X B}$. By uniqueness of such factorizations, $\eta_{A \times_X B} \circ f = Id_{L(A \times_X B)}$. So the unit is an equivalence and therefore the pullback is L-local. □

Corollary 4. For any reflective subuniverse, if X is local then so is the identity type $x = y$ for any two $x, y : X$.

Proof. We can obtain the identity type $x = y$ as the pullback of $x : 1 \to X$ and $y : 1 \to X$ for any $x, y : X$. So if X is modal we also have the identity type for any two terms of the type to be modal. □

Lemma 15. Given a reflective subuniverse, if $P(x)$ is local for all $x : X$, then so is $\Pi_{(x:X)}P(x)$.

Proof. By the same argument as the one we used before, it is suficient to define a left inverse to the modal unit $\eta : (\Pi_{(x:A)}P(x)) \to \bigcirc(\Pi_{(x:A)}P(x))$. Due to the universal property of the dependent products, finding the dotted arrow in the diagram

$$\begin{array}{ccc} \Pi_{(x:A)}P(x) & \xrightarrow{\quad id \quad} & \Pi_{(x:A)}P(x) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \theta :\equiv \lambda f.\lambda a.\eta_{P(a)}(f(a))} \\ L(\Pi_{(x:A)}P(x)) & \dashrightarrow & \Pi_{(x:A)}LP(x) \end{array}$$

is the same as finding it in

$$\begin{array}{ccc} \Pi_{(x:A)}P(x) & \xrightarrow{\quad ev_a \quad} & P(a) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ L(\Pi_{(x:A)}P(x)) & \dashrightarrow & LP(a) \end{array}$$

for any $a : A$, with $ev_a(f) :\equiv f(a) : (\Pi_{(x:A)}P(x)) \to P(x)$. Thus we take

$$f :\equiv \lambda m.\lambda a L(ev_a)(m) : L(\Pi_{(x:A)}P(x)) \to \Pi_{(x:A)}P(x)$$

as the solution to the first diagram. Now θ is an equivalence, since $P(x)$ is modal for any $x : A$. Therefore $f \circ \eta = Id$ and the unit for the dependent product is an equivalence. □

Then we have that if A and B are local, following the construction of the product as a dependent product $A \times B$ is also local. Moreover we have the following lemma.

Lemma 16. Every reflective subuniverse L is closed under finite cartesian products.

Proof. We have to show that the extension

$$\begin{array}{ccc} X \times Y & & \\ \downarrow{\scriptstyle \eta_{X\times Y}} & \searrow^{\varphi} & \\ L(X \times Y) & \dashrightarrow & LX \times LY \end{array}$$

is an equivalence,where $\varphi :\equiv \lambda(x, y).(\eta_X(x), \eta_Y(y))$. As $(L(X \times Y), \eta_{X\times Y}, -)$ is an inhabitant of the type in Lemma 12, it is enough to show that we can extend $(LX \times LY, \varphi)$ to an element of the type. Then the two elements will be equal inducing an equivalence that by uniqueness will be the one above. Currying the functions and using the equivalences, we get from Z being a local type the following equivalences:

$$\begin{aligned} (X \times Y \to Z) &\simeq X \to (Y \to Z) \\ &\simeq X \to (LY \to Z) \\ &\simeq LX \to (LY \to Z) \quad (\ LY \to Z \textit{ is modal due to the previous lemma}) \\ &\simeq LX \times LY \to Z \end{aligned}$$

Since $LX \times LY$ is modal and the equivalence is given by the precomposition we have the last element of the triple from the lemma and we have finished the proof. □

Lemma 17. Every reflective subuniverse L contains all mere propositions

Proof. It is almost direct to see that a type P is a mere proposition if and only if $P \to P \times P$ is an equivalence. And since any reflective subuniverse is closed under finite cartesian products, the result follows. □

Remark 12. Note that in general a reflective subuniverse will not be closed under Σ-types, not even when the indexing type is in the reflective subuniverse. Precisely when a reflective subuniverse is closed under dependent sums it has an induction principle and forms a modality,a property which we will be able to characterize in many ways.

3.2 Modalities in Homotopy Type Theory

As we already said we will deal only with idempotent monadic modalities. For convenience we will refer to those just as *modalities*. Modalities in homotopy type theory can be expressed as one of the following equivalent notions:

- Higher modalities
- Uniquely eliminating modalities
- Σ-closed reflective subuniverses
- Stable orthogonal factorization systems

All of these structures except for stable orthogonal factorization systems have in common the following: a **modal operator** $\bigcirc : \mathcal{U} \to \mathcal{U}$ and a **modal unit** $\eta^{\bigcirc} : \Pi_{A:\mathcal{U}} A \to \bigcirc A$. A type $X : \mathcal{U}$ will be modal if the restriction η^X of the unit to the type, is an equivalence. We will also denote $U_{\bigcirc} :\equiv \Sigma_{(X:\mathcal{U})} isModal(X)$, the *subuniverse of modal types*. The first of the definitions we will give of a modality is the one closest to the localizations we have already seen.

Definition 22. *A* reflective subuniverse *is a predicate* $isModal : \mathcal{U} \to \textbf{\textit{Prop}}$ *with a modal operator and a modal unit such that* $isModal(\bigcirc A)$ *and that for every* $B : \mathcal{U}$ *satisfying* $isModal(B)$ *then the function*

$$\lambda f. f \circ \eta_A : (\bigcirc A \to B) \to (A \to B)$$

is an equivalence. A reflective subuniverse is Σ*-closed if whenever* $isModal(X)$ *and* $isModal(P(x))$ *for all* $x : X$ *then we have* $isModal(\Sigma_{(x:X)} P(x))$.

Definition 23. *A* higher modality *is a modal operator and a modal unit together with:*

1. *For every type* $A : \mathcal{U}$ *and every dependent type* $P : \bigcirc A \to \mathcal{U}$*, a function*

$$ind^{\circ}_A : (\Pi_{(a:A)} \bigcirc P(\eta(a))) \to (\Pi_{(z:\bigcirc A)} \bigcirc (P(z)).$$

2. *The identification*

$$comp^{\circ}_A(f, x) : ind^{\circ}_A(f)(\eta(x)) = f(x)$$

for each, $x : A$ *and* $f : (\Pi_{(x:A)} \bigcirc (\eta(x)))$.

3. *For any* $x, y : A$*, the modal unit* $\eta_{x=y} : x = y \to \bigcirc(x = y)$ *is an equivalence.*

Definition 24. *A* uniquely eliminating modality *is a modal operator with a modal unit such that*

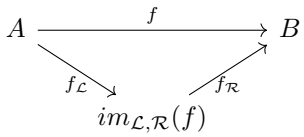
$$\lambda f. f \circ \eta_A : (\Pi_{(z:\neg\neg A)} \neg\neg(P(z)) \to (\Pi_{(x:A)} \neg\neg P(\eta(x))).$$

The last of these equivalent definitions is a stable orthogonal factorization system.

Definition 25. *An orthogonal factorization system consists of the predicates* $\mathcal{L}, \mathcal{R} : \Pi_{A,B:\mathcal{U}}(A \to B) \to Prop$ *such that*

1. *$\mathcal{L}$ and $\mathcal{R}$ are closed under composition and contain all identities (therefore forming subcategories of the category of types that contain all the objects);*

2. *the type $fact_{\mathcal{L},\mathcal{R}}(f)$ of factorizations of f*



is contractible. Being more precise, we define the type $fact_{\mathcal{L},\mathcal{R}}(f)$ as the one that has for terms the tuples

$$(im_{\mathcal{L},\mathcal{R}}(f), (f_{\mathcal{L}}, p), (f_{\mathcal{R}}, q), h)$$

where $f_{\mathcal{L}} : A \to im_{\mathcal{L},\mathcal{R}}(f)$ and $p : \mathcal{L}(f_{\mathcal{L}})$, $f_{\mathcal{R}} : im_{\mathcal{L},\mathcal{R}}(f) \to B$ and $q : \mathcal{R}(f_{\mathcal{R}})$, and finally an identification $h : f_{\mathcal{L}} \circ f_{\mathcal{R}} = f$. We call $im_{\mathcal{L},\mathcal{R}}(f)$ the $\mathcal{L},\mathcal{R} - image$ of f.

A type is said to be $\mathcal{L},\mathcal{R}$-modal if the map $!_{\mathcal{R}} : X \to 1$ is in $\mathcal{R}$, making $!_{\mathcal{L}}$ an equivalence [*]. We say that an orthogonal factorization system is stable if $\mathcal{L}$ is closed under pullback.

Remark 13. In a stable orthogonal factorization system both $\mathcal{L}$ and $\mathcal{R}$ are closed under pullback, but the later is closed indiferently of the property, as shown later in the section.

We have in each definition of a modality what it means for a type to be modal. As we have seen with reflective subuniverses, although in our lemma we did not require Σ-closedness, modalities are completely determined by the subuniverse $IsModal : \mathcal{U} \to \mathbf{Prop}$. These are Theorems 1.12,1.15,1.18, and 1.53 of [2].

This convenient property tells us that it is sufficient to show that any modality of one type determines one of the next type with the same modal types. We will begin by showing that a higher modality forms a uniquely eliminating modality, which at its turn induces a Σ-closed reflective universe. We will have to build up some theory for constrtucting of stable orthogonal factorization systems from reflective subuniverses, and finally we will see that a stable orthogonal system induces a higher modality.

3.2.1 Higher modalities

Theorem 14. *Every higher modality determines a uniquely eliminating modality with the same modal types.*

Proof. Let $\bigcirc$ be a modality with η_A as modal units. Since we already got the modal operator and unit, our goal is to show that the map

$$\lambda f.f \circ \eta_A : (\Pi_{(z:\bigcirc A)} \bigcirc (P(z)) \to (\Pi_{(x:A)} \bigcirc P(\eta(x)))$$

is an equivalence for $A : \mathcal{U}$ and $P : \bigcirc A \to \mathcal{U}$. The induction principle,ind°_A together with the computation rule $comp^{\circ}_A$ give us a right inverse of the precomposition map. To show that ind°_A is also a left inverse, consider a term $s : \Pi_{(z:\bigcirc A)} \bigcirc P(z)$. We need to find a homotopy

$$ind^{\circ}_A(s \circ \eta_A)(z) = \Pi_{(z:\bigcirc A)} s(z).$$

[*]Since there is only one function to $\mathbf{1}$, the contractibility of the tuple implies the contractibility (and thus equivalence) of the type in the left class.

We know by assumption that $P(x)$ is modal for each $z : \bigcirc A$, so it follows that $s(z) = ind^{\circ}_{A}(s \circ \eta_A)(z)$ for each $z : \bigcirc A$. Hence it is sufficient to find a function of type

$$ind^{\circ}_{A}(s \circ \eta_A)(\eta_A(a)) = \Pi_{(a:A)} s(\eta_A(z))$$

which we have by the computation rule of higher modalities. □

3.2.2 Uniquely eliminating modalities

Lemma 18. Given a uniquely eliminating modality, $\bigcirc A$ is modal for any type A.

Proof. Using the equivalence given in the definition of a uniquely eliminating modality we can find a function $f : \bigcirc \bigcirc X \to \bigcirc X$ and an identification $f \bigcirc \eta_{\bigcirc X} = id_{\bigcirc X}$. We also get that

$$(\bigcirc \bigcirc X \to \bigcirc \bigcirc X) \to (\bigcirc X \to \bigcirc \bigcirc X)$$

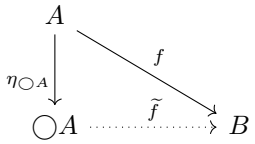
is an equivalence, hence its fiber over the unit $\eta_{\bigcirc X}$

$$\Sigma_{(g:\bigcirc\bigcirc X \to \bigcirc\bigcirc X)} g \circ \eta_{\bigcirc X} = \eta_{\bigcirc X}$$

is contractible. Since $id_{\bigcirc\bigcirc X}$ and $\eta_{\bigcirc X} \circ f$ are in the type, we get that f is also the right inverse of the unit, showing that $\eta_{\bigcirc X}$ is an equivalence and $\bigcirc X$ is modal. □

Theorem 15. *Any uniquely eliminating modality determines a Σ-closed reflective subuniverse.*

Proof. Fixing a constant P in the definition of uniquely eliminating modalities we can see that every map $f : A \to B$ into a modal type B has a homotpy unique extension to $\bigcirc A$ along the unit



Since the types $\bigcirc X$ are modal by the previous lemma, we get a reflective subuniverse. Now we have to check that the type $\Sigma_{z:\bigcirc X} \bigcirc (P(z))$ is modal for any type X and $P : X \to \mathcal{U}$. We have a function

$$\varphi :\equiv \lambda m.(f(m), g(m)) : \bigcirc(\sigma_{(z:\bigcirc X)} \bigcirc (P(z))) \to \Sigma_{(z:\bigcirc X)} \bigcirc (P(z))$$

where f and g also come defined by

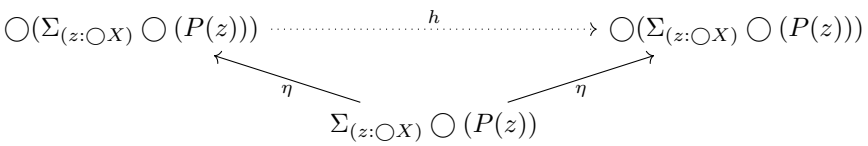
$$f :\equiv ind_{\circ}(\lambda x.\lambda u.x) : \bigcirc(\Sigma_{z:\bigcirc X)} \bigcirc (P(z))) \to \bigcirc X$$

$$f :\equiv ind_{\circ}(\lambda x.\lambda u.u) : \Pi_{w:\bigcirc(\Sigma_{z:\bigcirc X)} \bigcirc (P(f(w))}$$

We want to show that this φ is an inverse to the modal unit. Note that

$$\varphi(\eta(x, y)) = (f(\eta(x, y)), g(\eta(x, y))) = (x, y)$$

so that it is a left inverse of the unit is immediate. To show that φ is a right inverse of η, take the type of functions h such that the following diagram commutes :



It is a fiber over η of a precomposition equivalence, and therefore it is contractible. Since it also contains the identity, it is suficient to show that $\eta \circ \varphi \circ \eta = \eta$ to see that it is a right inverse, but since it is a left inverse we already know this, so φ is a left inverse of the modal unit. □

3.2.3 Σ-reflective subuniverses

Altough we have seen reflective subuniverses to have a *modal operator* and a *modal unit*, differently to higher modalities and uniquely eliminating modalities in this case we had *modal types* as part of the data given, we know though that a type $X : \mathcal{U}$ is L-local (in this case modal) if and only if the unit η_X is an equivalence. In this part we will introduce some theory that will help us prove that a Σ-closed reflective subuniverse produces a stable orthogonal factorization system. Namely we will introduce $\bigcirc$-connectedness, a generalization of the n-connectedness we defined in n-truncations.

Definition 26. *Let $M : \mathcal{U} \to Prop$ be a reflective subuniverse with modal operator $\bigcirc$. We say that a type $X : \mathcal{U}$ is $\bigcirc$-connected if $\bigcirc X$ is contractible, and we say that a function $f : X \to Y$ is $\bigcirc$-connected if the fibers are. Similarly we will say that f is* modal *if every one of the fibers is modal.*

Theorem 16. *Given a reflective subuniverse with modal operator $\bigcirc$, the following are equivalent:*

(1) It is Σ-closed.

(2) It is uniquely eliminating.

(3) The modal units are $\bigcirc$-connected.

Proof. We will leave this theorem unproven. The theorem with a proof can be found in [2] as Theorem 1.32 or in [9] as Lemma 7.5.7 and Theorem 7.7.4. □

Theorem 17. *Consider a Σ-closed reflective subuniverse with modal operator $\bigcirc$ and $P : X \to \mathcal{U}$ for some type $X : \mathcal{U}$. Then the unique map for which the diagram*

$$\begin{array}{ccc} \Sigma_{x:X} P(x) & & \\ \downarrow{\scriptstyle \eta} & \searrow^{\varphi} & \\ \bigcirc(\Sigma_{x:X} P(x)) & \dashrightarrow & \bigcirc(\Sigma_{x:X} \bigcirc (P(x))) \end{array}$$

with $\varphi :\equiv \lambda(x, y).\eta(x, \eta(y))$ commutes is an equivalence.

Proof. Since both codomains are modal it suffices to show that φ has the universal property of $\eta_{(\Sigma_{(x:X)} P(x))}$, i.e., that any map from $\Sigma_{x:X} P(x)$ to a modal type $Y : \mathcal{U}$ will extend uniquely to $\bigcirc(\Sigma_{x:X} \bigcirc (P(x)))$.

$$\begin{aligned} \Sigma_{x:X} P(x) \to Y &\simeq \Pi_{(x:X)} (P(x) \to Y)^* \\ &\simeq \Pi_{(x:X)} \bigcirc (P(x)) \to Y \\ &\simeq (\Sigma_{(x:X)} \bigcirc (P(x))) \to Y \\ &\simeq \bigcirc(\Sigma_{(x:X)} \bigcirc (P(x))) \to Y \end{aligned}$$

so we have the desired map. □

Theorem 18. *Given $f : A \to B$ and $g : B \to C$ and a reflective subuniverse $\bigcirc$, if f is $\bigcirc$-connected, then g is $\bigcirc$-connected if and only if $g \circ f$ is $\bigcirc$-connected.*

Proof. We have $fib_{g \circ f}(z) = \Sigma_{p:fib_g(z)} fib_f(pr_1(p))$. Then for any $z : C$ we have:

$$\begin{aligned} \bigcirc(fib_{g \circ f}(z)) &\simeq \bigcirc(\Sigma_{p:fib_g(z)} fib_f(pr_1(p))) \\ &\simeq \bigcirc(\Sigma_{p:fib_g(z)} \bigcirc (fib_f(pr_1(p)))) \\ &\simeq \bigcirc(\Sigma_{p:fib_g(z)} \mathbf{1}) \\ &\simeq \bigcirc(fib_g(z)) \end{aligned}$$

using the fact that f is $\bigcirc$-connected and the previous lemma. Therefore one is contractible if and only if the other is. □

Remark 19. The properties we are seeing now resemble those of n-truncations since these are a specific case of a Σ-closed reflective subuniverse. But most of what we have seen for modalities can also be shown for reflective universes not necessarily closed for sigma types.

Example 7. Let P be a mere proposition. The *closed modality* determined by P is the one taking $A : \mathcal{U}$ to $Cl_P(A) :\equiv A * P$ (the pushout of Q and A under $A \times Q$). We show that this is a Σ-closed reflective subuniverse. First we define a type B to be modal if $P \to IsContr(B)$. Note that $P \to IsContr(B * P)$ is always true. By the universal property of pushouts for a map $P * A \to B$, we have a map $f : A \to B$ and $g : P \to B$ together with an identification $e : f(a) = g(p)$, but if $P \to isContr(B)$, then g, e are uniquely determined and this data is just the map $f : A \to B$. Thus

$$(Cl_P A \to B) \to (A \to B)$$

is an equivalence. Moreover, the type of dependent pairs, of a contractible family indexed over a contractible type is at its time contractible, so the reflective subuniverse is closed under Σ-types

Theorem 20. *A Σ-closed reflective subuniverse determines a stable orthogonal factorization system with the same modal types.*

Proof. We will define $\mathcal{L}$ as the class of ◯-connected maps and $\mathcal{R}$ as the class of modal maps. We first show that both classes are closed under composition and contain all the identities, the first property in the definition of stable orthogonal factorization systems. Since $fib_{g\circ f}(z) = \Sigma_{p:fib_g(z)} fib_f(pr_1(p))$, by the Σ-closedness of the reflective subuniverse, if f and g are in $\mathcal{R}$, so is the composition $g \circ f$. The previous lemma says exactly that $\mathcal{L}$, the class of ◯-connected maps, is closed under composition. Now, the fibers of an identity map are contractible, and contractible types are both modal and ◯-connected so we have all the identities in both $\mathcal{L}$ and $\mathcal{R}$.

It remains to show that the type of factorizations for any map $f : X \to Y$ is contractible. Since

$$(X, f) =_{(\Sigma_{Z:\mathcal{U}} Z \to Y)} (\Sigma_{(y:Y)} fib_f(y), pr_1)$$

it is suficient to show that $fact_{\mathcal{L},\mathcal{R}}(pr_1)$ is contractible for any $pr_1 : \Sigma_{(y:Y)} P(y) \to Y$. This factors as

$$\Sigma_{(y:Y)} P(y) \underset{p_{\mathcal{L}}}{\longrightarrow} \Sigma_{(y:Y)} \bigcirc (P(y)) \underset{p_{\mathcal{R}}}{\longrightarrow} Y$$

where $p_{\mathcal{L}} :\equiv total(\eta_{P(-)})$ and $p_{\mathcal{R}} :\equiv pr_1$, where $total(f)$ for a map $f : \Pi_{(a:A)} P(a) \to Q(a)$ is the *induced map on total spaces*:

$$total(f) :\equiv \lambda w.(pr_1(w), f(pr_1(w), pr_2(w))) : \Sigma_{a:A} P(a) \to \Sigma_{(a:A)} Q(a).$$

The fibers of $p_{\mathcal{R}}$ are of the form $\bigcirc(P(-))$, so it is clearly in $\mathcal{R}$. Moreover $fib_{total(\eta(-))}((y, u)) = fib_{\eta_{P(y)}}(u)$ and each η is ◯-connected, therefore $p_{\mathcal{L}}$ is in $\mathcal{L}$.

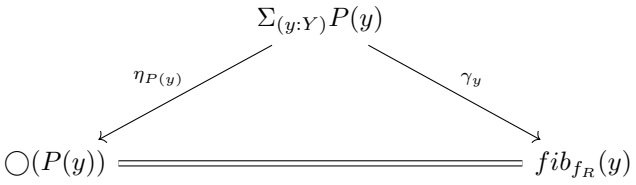
If we consider any other factorization (Im, f_L, f_R, h) of pr_1 into $f_L : (\Sigma_{(y:Y)} P(y)) \to Im$ followed by $f_R : IM \to Y$ an $\mathcal{R}$-map, since $Im = \Sigma_{(y:Y) fib_{f_R}(y)}$ we get the following square :

$$\begin{array}{ccc} \Sigma_{(y:Y)} P(y) & \xrightarrow{f_L} & Im \\ {\scriptstyle total(\gamma)}\downarrow & \nearrow & \downarrow{\scriptstyle f_R} \\ \Sigma_{(y:Y)} fib_{f_R}(y) & \xrightarrow{pr_1} & Y \end{array}$$

where $\gamma(y, u) :\equiv (f_L(y, u), h(y, u))$. It follows then the equality

$$(Im, f_L, f_R, h) = (\Sigma_{(y:Y) fib_{f_R}(y)}, total(\gamma), pr_1, -).$$

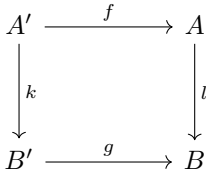
Therefore it suffices to show that there is a triangle such that for any $y : Y$



commutes. For this it is enough to show that $\eta_{P(y)}$ and γ_y have the same universal property, deriving a homotopy given by precomposition

$$\begin{aligned}(fib_{f_R}(Y) \to Z &\simeq ((\Sigma_{(w:fib_{f_R}(y))} \bigcirc (fib_{f_L}(pr_1(w)))) \to Z \\ &\simeq ((\Sigma_{(w:fib_{f_R}(y))}(fib_{f_L}(pr_1(w)))) \to Z \\ &\simeq (fib_{f_R \circ f_L}(y)) \to Z \\ &\simeq P(y) \to Z.\end{aligned}$$

Finally it remains to be shown that the orthogonal factorization system we have constructed is stable. Take a pullback diagram



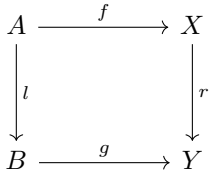
in which l is in $\mathcal{L}$. By the *pasting lemma for pullbacks*, it follows that $fib_k(b) = fib_l(g(b))$ for each $b : B'$. Therefore k is in $\mathcal{L}$. □

3.2.4 Orthogonal factorization systems

In this section we will develop some of the theory of orthogonal factorization systems for its use in the definition and properties of *stable* orthogonal factorization systems. We will not give some of the proofs in this chapter since they are long and require some technical tools. The proofs can be found mostly in [12], in Chapter 7.

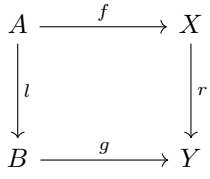
In category theory an orthogonal factorization system is characterized by an equivalent lifting property. We will show now the analogue of it in our context.

Definition 27. *Let $(\mathcal{R}, \mathcal{L})$ be an orthogonal factorization system and consider a commutative square*



with l in $\mathcal{L}$ and r in $\mathcal{R}$. We take $fill(S)$ (with $S : r \circ f = g \circ l$) to be the type of diagonal fillers, the tuple (j, H_f, H_g, K) where $j : B \to X$, $H_f : j \circ l = f$, $H_g : r \circ j = g$ and $K : r \circ H_f = S \cdot (H_g \circ l)$. The last equality is required for homotopy coherence, since the square and triangles commuting are not mere propositions but homotopies inhabiting these types.

Lemma 19. Let $(\mathcal{R}, \mathcal{L})$ be an orthogonal factorization system, and let the commutative square



with l in $\mathcal{L}$ and r in $\mathcal{R}$. Then the type $fill(S)$ of diagonal fillers is contractible.

Definition 28. *For a class $\mathcal{C} : \Pi_{A,B:\mathcal{U}}(A \to B) \to$ **Prop** of maps we define:*

1. *${}^{\perp}\mathcal{C}$ to be the class of maps with the* unique lifting property *with respect to all maps in $\mathcal{C}$. The mere proposition ${}^{\perp}\mathcal{C}(l)$ asserts that for every commutative square*

$$\begin{array}{ccc} A & \xrightarrow{f} & X \\ \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle r} \\ B & \xrightarrow{g} & Y \end{array}$$

   with r in $\mathcal{C}$, the type $fill(S)$ is contractible.

2. *$\mathcal{C}^{\perp}$ to be the class of maps with the* unique lifting property *with respect to all maps in $\mathcal{C}$.*

3. *$l \perp r$ is the same as $r \in l^{\perp}$.*

Lemma 20. Let $(\mathcal{R}, \mathcal{L})$ be an orthogonal factorization system. Then $\mathcal{L} =^{\perp} \mathcal{R}$ and $\mathcal{L}^{\perp} = \mathcal{R}$

Proof. We show that $\mathcal{L} =^{\perp} \mathcal{R}$, and the second equality can be proved the same way. Since for any map f the implication $\mathcal{L}(f) \to^{\perp} \mathcal{R}(f)$ follows from Lemma 19, we only need to prove ${}^{\perp}\mathcal{R}(f) \to \mathcal{L}(f)$.

Take a map $f : A \to B$ in ${}^{\perp}\mathcal{R}$, and it's factorization (f_L, f_R). Then the diagram

$$\begin{array}{ccc} A & \xrightarrow{f_R} & Im_{\mathcal{L},\mathcal{R}}(f) \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f_R} \\ B & \xrightarrow{id} & B \end{array}$$

commutes. Since f has the left lifting propert, the type of diagonall fillers of the diagram is contractible and we have a section j of f_R. The map $j \circ f_R$ is a diagonal filler of the square

$$\begin{array}{ccc} A & \xrightarrow{f_L} & Im_{\mathcal{L},\mathcal{R}}(f) \\ \downarrow{\scriptstyle f_L} & & \downarrow{\scriptstyle f_R} \\ Im_{\mathcal{L},\mathcal{R}}(f) & \xrightarrow{f_R} & B \end{array}$$

The identity map for the type $Im_{\mathcal{L},\mathcal{R}}(f)$ is also a diagonal filler so since the type is contractible $j \circ f_R = Id$ thus j and f_R are inverse equivalences and $(B, f) = (Im_{\mathcal{L},\mathcal{R}}(f), f_L)$, hence f is in $\mathcal{L}$ since f_L is. □

Corollary 5. The data of two orthogonal factorization systems $(\mathcal{R}, \mathcal{L})$ and $(\mathcal{R}', \mathcal{L}')$ are identical if and only if $\mathcal{R} = \mathcal{R}'$.

Proof. The only if implication is trivial. If $\mathcal{R} = \mathcal{R}'$, by the previous lemma we have $\mathcal{L}$ and $\mathcal{L}'$ to be the class of maps with the unique right lifting property with respect to $\mathcal{R} = \mathcal{R}'$, so they are equal and the rest of the data constituting an orthogonal factorization system is a mere proposition. □

Lemma 21. Let $(\mathcal{R}, \mathcal{L})$ be an orthogonal factorization system. Then the class $\mathcal{R}$ is stable under pullbacks.

Proof. Consider the pullback

$$\begin{array}{ccc} A & \xrightarrow{g} & X \\ \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle h} \\ B & \xrightarrow{f} & Y \end{array}$$

with $h : X \to Y$ in $\mathcal{R}$, and $k = k_R \circ k_L$ a factorization of k Then the outer rectangle in

$$\begin{array}{ccccc} A & = & A & \xrightarrow{g} & X \\ \downarrow{\scriptstyle k_L} & & \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle h} \\ Im_{\mathcal{L},\mathcal{R}}(k) & \xrightarrow{g} & B & \xrightarrow{f} & Y \end{array}$$

commutes, therefore there is a diagonal lift $i : Im_{\mathcal{L},\mathcal{R}}(k) \to X$ with $i \circ k_L = g$ and $h \circ i = f \circ k_R$. Then by the universal property of the pullback we get $j : Im_{\mathcal{L},\mathcal{R}}(k) \to A$ with $g \circ j = i$ and $f \circ j = k_R$. Then as we have

$$g \circ j \circ k_L = i \circ k_L = g$$
$$k \circ j \circ k_L = k_R \circ k_L = k$$

we get by uniqueness that $j \circ k_L = id$. To show that k_L is an equivalence it remains to show that $k_L \circ j = Id$. Consider the square

$$\begin{array}{ccc} A & \xrightarrow{k_L} & Im_{\mathcal{L},\mathcal{R}}(k) \\ \downarrow{\scriptstyle k_L} & & \downarrow{\scriptstyle k_R} \\ Im_{\mathcal{L},\mathcal{R}}(k) & \xrightarrow{k_R} & B \end{array}$$

for which the identity is the filler and the homotopies making the triangles commute are trivial. Since we have the homotopies $k_L \circ j \circ k_R \sim k_L$ and $k_R \circ k_L \circ j \sim k_R$ we get another diagonal filler $k_L \circ j$, since the type of diagonal fillers is contractible, $k_L \circ j = Id$. □

Stable orthogonal factorization systems

Before proving that a stable orthogonal factorization system will induce a higher modality we introduce some helping lemmas.

Lemma 22. Given l, r, f, g and a homotpy $S : r \circ f = g \circ l$, consider as $b : B$ varies, all the diagrams of the form

$$\begin{array}{ccccc} fib_l(b) & \xrightarrow{pr_1} & A & \xrightarrow{f} & X \\ \downarrow{\scriptstyle !} & & \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle r} \\ 1 & \xrightarrow{b} & B & \xrightarrow{g} & Y \end{array}$$

and write $S_b : r \circ (f \circ pr_1) = (g \circ b) \circ !$ for the induced commutative square. Then the map defined by the precomposition with b

$$fill(S) \to \Pi_{(b:B)} fill(S_b)$$

is an equivalence.

As corollary to this lemma we have the following two facts for any stable orthogonal factorization system:

- If l is a map such that $fib_l(b) \to \mathbf{1}$ is in $\mathcal{L}$ for each $b : B$ (with B the domain), then l is also in $\mathcal{L}$.
- if $l \perp r$ for all maps $l : A \to \mathbf{1}$, then r is in $\mathcal{R}$. In particular for any modality $\bigcirc$, if $X \to (A \to X)$ is an equivalence for all $\bigcirc$-connected types then X is modal.

Lemma 23. Let $(\mathcal{R}, \mathcal{L})$ be a stable orthogonal factorization system. Then a map $r : X \to Y$ is in $\mathcal{R}$ if and only if $fib_r(y)$ is $(\mathcal{R}, \mathcal{L})$-modal for every $y : Y$.

Proof. It is sufficient to show that any map with modal fibers is in $\mathcal{R}$, since we already know $\mathcal{R}$ is closed under pullbacks. So let $r : X \to Y$ be a map whose fibers are modal. Since we have $\mathcal{L} =^{\perp} \mathcal{R}$ it suffices to show that r has the right lifting property with respect to $\mathcal{L}$. Consider a diagram

$$\begin{array}{ccc} A & \xrightarrow{f} & X \\ \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle r} \\ B & \xrightarrow{g} & Y \end{array}$$

where l is in $\mathcal{L}$. We would want to show then that the type of diagonal fillers is contractible. By Lemma 22 the type of diagonal fillers is equivalent to the one of the product of the fiagonal fillers indexed by $b : B$ of the following squares:

$$\begin{array}{ccc} fib_l(b) & \xrightarrow{f \circ i_b} & X \\ \downarrow & & \downarrow{\scriptstyle r} \\ \mathbf{1} & \xrightarrow{g(b)} & Y \end{array}$$

Therefore it suffices to show contractibility for each $b : B$. Since the fillers factor through the pullback $\mathbf{1} \times_Y X$ which is $fib_l(g(b))$, the type of the diagonal fillers of the previous diagram is

$$\begin{array}{ccc} fib_l(b) & \dashrightarrow & fib_r(g(b)) \\ \downarrow & & \downarrow \\ \mathbf{1} & = & Y \end{array}$$

where the map above is the unique map into the pullback $fib_r(g(b))$. Since by assumption we are working in a stable orthogonal factorization system the left map down is in $\mathcal{L}$ by stability, and the right map is in the right class by assumption, so the type of diagonal fillers is contractible. □

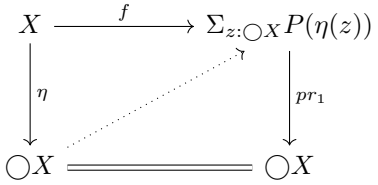
Theorem 21. *Any stable orthogonal factorization system determines a higher modality with the same modal types.*

Proof. For every type X we have the unique factorization $X \to \bigcirc X \to \mathbf{1}$ of the unique map $X \to \mathbf{1}$. This determines the modal unit $\eta : X \to \bigcirc X$ which is in $\mathcal{L}$, and the unique map $\bigcirc X \to 1$ is in $\mathcal{R}$, therefore $\bigcirc X$ is $(\mathcal{R}, \mathcal{L})$-modal.

To get the induction principle of the higher modality, let $P : \bigcirc X \to U$ and $f : \Pi_{x:X} \bigcirc P(\eta(x)))$. We have the square

$$\begin{array}{ccc} X & \xrightarrow{f} & \Sigma_{z:\bigcirc X} P(\eta(z)) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle pr_1} \\ \bigcirc X & = & \bigcirc X \end{array}$$

Note that pr_1 is in $\mathcal{R}$ because its fibers are modal and the modal unit η is in $\mathcal{L}$. Therefore as we have seen in the definition of the unique lifting properties, the type of fillers of this square is contractible. So we get



with $s : X \to \Sigma_{z:\bigcirc X} P(\eta(z))$ and homotopies making the two triangles commute, whose composite is the following type:

$$\Sigma_{(s:\bigcirc X \to \Sigma_{(z:\bigcirc X)} \bigcirc(P(z)))} \Sigma_{(H:\Pi_{(z:\bigcirc X)} pr_1(s(z)=z))} \Sigma_{(K:\Pi_{(x:X)} s(\eta(x))=f(x))} \Pi_{(x:X)} pr_1(K(x)) = H(\eta(x)).$$

We can decompose s,f and K and rewrite the type as,

$$\Sigma_{(s_1:\bigcirc X \to \bigcirc X} \Sigma_{(s_2:\Pi_{(z:\bigcirc X)} \bigcirc(P(z)))} \Sigma_{(H:\Pi_{(z:\bigcirc X)} s(z)=z)} \quad \Sigma_{(K_1:\Pi_{(x:X)} s(\eta(x))=f_1(x))} \Sigma_{(K_2:\Pi_{(x:X)} s(\eta(x))=_{K_1(x)} f_2(x))} \quad \Pi_{(x:X)} K_1(x) = H(\eta(x))$$

Finally we can contract s_1, H and K_1 :

$$\Sigma_{(s_2:\Pi_{(z:\bigcirc X)} \bigcirc(P(z)))} \Pi_{(x:X)} s_2(\eta(x)) = f_2(x)$$

getting the type of extensions of f along η, that is, the fiber of the precomposition by η. Thus, this same precomposition is an equivalence so we have a uniquely eliminating modality, for which the identity types for $\bigcirc X$ are modal, so we get a higher modality. □

Chapter 4

Accessible Reflective Subuniverses

In category theory *localization* is the action of adding formal inverses to a class of maps F in a category $\mathcal{C}$, obtaining a category $\mathcal{C}[F^{-1}]$ and a functor $\mathcal{C} \to \mathcal{C}[F^{-1}]$ characterized by a universal property. We will give in this section an analogous of this construction, by giving reflective subuniverses of F-local objects.

4.1 Localizing at maps

Definition 29. *Given a family $F : \Pi_{(a:A)}B(a) \to C(a)$ of maps, a type $X : \mathcal{U}$ is F-local if*

$$\lambda g.g \circ F_a : (C(a) \to X) \to (B(a) \to X)$$

is an equivalence for each $a : A$.

A recurrent instance of this definition in this section will be that of a family $F : \Pi_{(a:A)}B(a) \to \mathbf{1}$, which we call a B-nullification.

Definition 30. *For a family $B : A \to \mathcal{U}$, a type $X : \mathcal{U}$ is B-*null *if*

$$\lambda x \lambda b.x : X \to (B(a) \to X)$$

is an equivalence for each $a : A$.

Remark 22. We can also think about F-local types as the ones that see each map in the family F as equivalences, and, specializing to nullifications, B-null types are the ones that see $B(a)$ as contractible types.

Next we will introduce the localization operator and show that a reflective subuniverse arises from it. We define the modal operator $\mathcal{L}_F : \mathcal{U} \to \mathcal{U}$ called the *localization at F*. The construction will be as a higher inductive type, using one of the constructors as the modal unit and the rest to build the data making the precompositions at F_a equivalences. Now let $F : \Pi_{(a:A)}B(a) \to C(a)$ be a family of maps and let $\mathcal{J}_F(X)$ be the inductive type with the following constructors:

- $\alpha_X : X \to \mathcal{J}_F(X)$
- $ext : \Pi_{\{a:A\}}(B(a) \to \mathcal{J}_F(X)) \to (C(a) \to \mathcal{J}_F(X))$
- $isext : \Pi_{\{a:A\}}\Pi_{(B(a)\to\mathcal{J}_F(X))}\Pi_{(b:B(a))}ext(f)(F_a(b)) = f(a)$

Being the inductive principle of $\mathcal{J}_F(X)$ is that for any family $P : \mathcal{J}_F(X) \to \mathcal{U}'$ if there are types

$$\begin{aligned}
&N : \Pi_{(x:X)}P(\alpha_X(x)) \\
&R : \Pi_{\{a:A\}}\Pi_{(f:B(a)\to\mathcal{J}_F(X))}(\Pi_{b:B(a)}P(f(b))) \to \Pi_{(c:C(a))}P(ext(f,c)) \\
&S : \Pi_{\{a:A\}}\Pi_{(f:B(a)\to\mathcal{J}_F(X))}\Pi_{f':\Pi_{b:B(a)}P(f(b))}\Pi_{b:B(a)}R(f')(F_a(b)) =^P_{isext(f,b)} f'(b)
\end{aligned}$$

Then there is a section $s : \Pi_{(x:\mathcal{J}_F(X)}P(X)$ such that $s \circ \alpha_X = N$. Note that the family P can land in a universe $\mathcal{U}'$, not necessarily $\mathcal{U}$. We will see that this higher inductive type behaves as we would want for the localization at X.

Lemma 24. If Y is F-local, then the precomposition with α_X

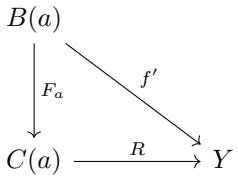
$$(- \circ \alpha_X) : (\mathcal{J}_F(X) \to Y) \to (X \to Y)$$

is an equivalence.

Proof. To show the equivalence we will show that this map is path-split. First we construct a right inverse to it: given a $g : X \to Y$ we have to extend it to $\mathcal{J}_F(X)$. We can apply the induction principle using the constant family Y over $\mathcal{J}_F(X)$ and $N :\equiv g$, so that the computation rule shows that what we get is an extension of g. To construct R and S, we take

$$f : B(a) \to \mathcal{J}_F(X)$$
$$f : B(a) \to Y$$

and we want to construct $R(f, f') : C(a) \to Y$ together with a witness $S(f, f')$ of the following triangle commuting



Since Y is F-local, the map

$$(- \circ F_a) : (C(a) \to Y) \to (B(a) \to Y)$$

is an equivalence, and therefore it has a right inverse, which applied to f' gives R and S. The other part of seeing that the map is path-split is to construct a right inverse to

$$ap_{(-\circ\alpha_X)} : (g = h) \to g \circ \alpha_X = h \circ \alpha_X$$

given $g, h : \mathcal{J}_F(X) \to Y$.

Suppose we have $K : \Pi_{(x:X)} g(\alpha_X(x)) = h(\alpha_X(x))$. We have to extend K to a homotopy $K' : \Pi_{(j:\mathcal{J}_F(X))} g(j) = h(j)$ such that $K'(\alpha_X(x)) = K(x)$. We will use the induction principle for the family given by $P(z) :\equiv (g(z) = h(z))$ and $N :\equiv K$. Again we have to construct *and*, let $f : B(a) \to \mathcal{J}_F(X)$ and $f' : \Pi_{b:B(a)} g(f(b)) = h(f(b))$. We want to get $R(f, f') : \Pi_{c:C(a)} g(ext(f, c)) = h(ext(f, c))$ and $S(f, f')$ a witness of

$$R(f, f')(F_a(b)) = g(isext(f, b) \cdot f'(b) \cdot h(isext(f, b))^{-1}$$

for any $b : B(a)$. Once again, since Y is F-local the map

$$(- \circ F_a) : (C(a) \to Y) \to (B(a) \to Y)$$

is an equivalence and in particular

$$ap_{(-\circ F_a)} : (g \circ ext(f, b) = h \circ ext(f, b)) \to (g \circ ext(f, b) \circ F_a = h \circ ext(f, b) \circ F_a)$$

has a right inverse, and since the right part of $R(f, f')(F_a(b)) = g(isext(f, b) \cdot f'(b) \cdot h(isext(f, b))^{-1}$ inhabits $(g \circ ext(f, b) \circ F_a = h \circ ext(f, b) \circ F_a)$, applying the right inverse we get R and S. □

Since the constructors give a right inverse to the map

$$(- \circ F_a) : (C(a) \to Y) \to (B(a) \to Y)$$

but do not make it an equivalence, in general $\mathcal{J}_F(X)$ is not an F-local type.

Definition 31. *Given any map $f : A \to B$, let*

$$\triangle_f : B \to A \times_B A$$
$$\triangledown_f : B +_A B \to B$$

be respectively the diagonal and the codiagonal of f.

Lemma 25. For any $f : B \to C$ and any X, we have a commuting diagram

$$\begin{array}{ccc} & (C \to X) & \\ {\scriptstyle (-\circ\triangledown_f)}\swarrow & & \searrow{\scriptstyle \triangle_{(-\circ f)}} \\ (C +_B C \to X) & \longrightarrow & (C \to X) \times_{(B\to X)} (C \to X) \end{array}$$

in which the bottom map is an equivalence.

Lemma 26. For any $f : B \to C$ we have the following equivalences:

$$isEquiv(f) \simeq pathsplit(f) \simeq rinv(f) \times rinv(\triangledown_f).$$

We can use these two properties of diagonal maps to devise a way in which we can make our localization work:

Lemma 27. For $f : B \to C$, a type Y is f-local if and only if both maps

$$(- \circ f) : (C \to Y) \to (B \to Y)$$
$$(- \circ \triangledown_f) : (C \to Y) \to (C +_B C \to Y)$$

have right inverses and if and only if both of these maps are equivalences.

Proof. By 26, our Y is f-local if and only if both precomposition with f and $\triangle_{-\circ f}$ have a right inverse. But the latter is equivalent to $(- \circ \triangledown_f)$ by 25. The second statement follows from the diagonal of an equivalence being an equivalence. □

Therefore to localize at F we only have to add the maps $\triangledown_{F_a}$ to F.

Definition 32. *For any $X : \mathcal{U}$, the localization of X at a family F is $\mathcal{L}_F(X) :\equiv \mathcal{J}_{\hat{F}}(X)$ and $\eta_X : X \to \mathcal{L}_F(X)$ is $\alpha^{\hat{F}}_X$, where*

$$\hat{F} : \Pi_{(a:A+A)} \hat{B}(a) \to \hat{C}(a)$$

and in the left we have

$$\hat{B}(inl(a)) :\equiv B(a) \quad \hat{C}(inl(a)) :\equiv C(a) \quad \hat{F}(inl(a)) :\equiv f_a$$

and in the right

$$\hat{B}(inr(a)) :\equiv C(a) +_{B(a)} C(a) \quad \hat{C}(inr(a)) :\equiv C(a) \quad \hat{F}(inr(a)) :\equiv \triangledown_{f_a}.$$

Theorem 23. *The subuniverse of F-local types forms a reflective subuniverse.*

Proof. We first have to show that for any $F : \Pi_{(a:A)} B(a) \to C(a)$, the types of the form $\mathcal{L}_F(X)$ are F-local. The constructors of the higher inductive type say that the maps

$$(- \circ \hat{F}_a) : (\hat{C}(a) \to \mathcal{J}_{\hat{F}}(X)) \to (\hat{B}(a) \to \mathcal{J}_{\hat{F}}(X))$$

have right inverses for all $a : A + A$. By definition of $\hat{F}$ these maps are the precomposition with F_a and $\triangledown_{F_a}$ therefore $\mathcal{J}_{\hat{F}}(X)$ is F-local.

That the precomposition with η_X follows from the fact that any F-local type is also $\hat{F}$-local and we have shown that then the precomposition with α_X is an equivalence. □

4.2 Accessible Modalities

We call a modality *accessible* if it can be generated by localization at a family of maps. Accessible modalities include all the examples we have seen until now, and we can actually check that there is a correspondence between *modalities* and a specific class of localizations which we called *nullifications*.

4.2.1 Nullification

Theorem 24. *If $F : \Pi_{a:A} B(a) \to C(a)$ is such that $C(a) = \mathbf{1}$ for each $a : A$, then the localization at F is a modality, the* nullification at B.

Proof. Since in the previous section we have seen that the localization at a family of maps forms a reflective subuniverse, it is enough to see that for the family $B : A \to \mathcal{U}$, the B-null types are Σ-closed. Then for any $a : A$ we need an equivalence

$$\Sigma_{(x:X)} Y(x) \to (B(a) \to \Sigma_{(x:X)} Y(x)).$$

We have the following equivalences:

$$\begin{aligned} (B(a) \to \Sigma_{(x:X)} Y(x)) &\simeq \Sigma_{(g:B(a)\to X)} \Pi_{(b:B(a))} Y(g(b)) \\ &\simeq \Sigma_{(x:X)} B(a) \to Y(x) \\ &\simeq \Sigma_{(x:X)} Y(x) \end{aligned}$$

Thus $\Sigma_{(x:X)} Y(x)$ is B-null. □

The converse is not true: if an F-localization is a modality the maps are not necessarily a null family, but we have the following:

Theorem 25. *If $F : \Pi_{(a:A)} B(a) \to C(a)$ is such that L_F is a modality, there is a family $E : I \to \mathcal{U}$ such that L_F coincides with the E-nullification.*

Proof. We write L_F for the reflector and η for its modal unit. We define $I = \Sigma_{(a:A)}(L_F(B(a)) + L_F(C(a)))$ and E by

$$\begin{aligned} E(a, inl(b)) &= fib_{\eta_{B(a)}}(b) \\ E(a, inl(c)) &= fib_{\eta_{C(a)}}(c) \end{aligned}$$

Since η is L_F-connected so is each $E(i)$, therefore an F-local type is E-null. Suppose now that X is an E-null type. Each $\eta_{B(a)}$ and $\eta_{C(a)}$ are L_E-connected (since the fibers are L_E-connected), thus X is $\eta_{B(a)}$-local and $\eta_{C(a)}$-local. Then we have a commutative square

$$\begin{array}{ccc} B(a) & \xrightarrow{\eta_{B(a)}} & L_F(B(a)) \\ {\scriptstyle F_a}\downarrow & & \downarrow{\scriptstyle L_F(F_a)} \\ C(a) & \xrightarrow{\eta_{C(a)}} & L_F(C(a)) \end{array}$$

and $L_F(F_a)$ is an equivalence so X is F_a-local. □

We can then give the following definitions :

Definition 33. *A reflective subuniverse on $\mathcal{U}$ is said to be* accessible *if it is the localization at a family of maps in $\mathcal{U}$.*

Definition 34. *A modality $\bigcirc$ is said to be an* accessible modality *if it is the nullification at a family $B : A \to \mathcal{U}$.*

Definition 35. *A* presentation *of a reflective subuniverse $\bigcirc$ is a family $F : \Pi_{(a:A)} B(a) \to C(a)$ with $A : \mathcal{U}$ $B, C : A \to \mathcal{U}$ such that $\bigcirc :\equiv \mathcal{L}_F$. In the case of a modality the family C is constant at the unit type.*

Example 8. The *closed modality* at a mere propositon P from the example on Σ-closed reflective subunivserses is presented by the nullification at the type family $\lambda x.\mathbf{0} : P \to \mathcal{U}$. A type A is null for this family if and only if $A \to (0 \to A)$ is an equivalence for any $p : P$, but $\mathbf{0} \to P$ is contractible, so being A-null is the same as $P \to IsContr(A)$, which is the definition of modal types for the closed modality.

4.2.2 Universe hierarchies and accessible extension

Recall that a universe is a type whose terms are itself types. As we would in set theory, we might want a universe $\mathcal{U}$ that contains all types, including itself. But as it happens in set theory this is not appropriate, as we can deduce from it that all types (including the empty type that represents the proposition $False$), are inhabited. Furthermore, we could encode Russell's paradox.
To avoid this we introduce a hierarchy of universes:

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \mathcal{U}_3 : \dots$$

where every universe $\mathcal{U}_n$ is a term of the next universe $\mathcal{U}_{n+1}$. We also assume that these universes are cumulative, that is, the elements of the last universe are also elements of the next one. So when we say that A is a type we mean that it inhabits some universe $\mathcal{U}_i$. As we have been doing we omit the level of the universes when stating a type, leaving the indices implicit. When some universe $\mathcal{U}$ is assumed, we refer to types belonging to $\mathcal{U}$ as small.

One use of accessibility is moving modalities through universes. Although our definitions of modalities and reflective subuniverses are bounded to a particular universe $\mathcal{U}$, most of them are "uniform" or "polymorphic" definitions, that is, they apply to types indifferently of their universe. Accessibility will ensure that this holds and moreover that these *modal operators* in different universes match in an appropriate way.

Theorem 26. *If $\bigcirc$ is an accessible reflective subuniverse defined on a universe $\mathcal{U}$ and $\mathcal{U}'$ is a universe such that $\mathcal{U} : \mathcal{U}'$ then there is a reflective subuniverse $\bigcirc'$ on $\mathcal{U}'$ such that:*

(i) If $\bigcirc$ is a modality so is $\bigcirc'$.

(ii) A type $X : \mathcal{U}$ is $\bigcirc'$-modal if and only if it is $\bigcirc$-modal.

(iii) For a type $X : \mathcal{U}$ the induced map $\bigcirc' X \to \bigcirc X$ is an equivalence.

(iv) A type $X : \mathcal{U}'$ is $\bigcirc'$-modal if and only if $(- \circ f) : (B \to X) \to (A \to X)$ is an equivalence for any map $f : A \to B$ in $\mathcal{U}$ such that $\bigcirc(f)$ is an equivalence .

(v) $\bigcirc'$ depends only on $\bigcirc$, not on its presentation.

Proof. We know that $\bigcirc$ is generated by some family $F : \Pi_{(a:A)} B(a) \to C(a)$. We define $\bigcirc' : \mathcal{U}' \to \mathcal{U}'$ to be the *higher inductive localization* at the same family F which will be in $\mathcal{U}'$ since $\mathcal{U} : \mathcal{U}'$ and the bigger universe will contain the types of the previous ones.

(i) If $\bigcirc$ is a modality we can take the family with $C(a) = 1$, so $\bigcirc'$ will also be a modality.

(ii) The notion of a type being $F - local$ is independent of the universe the type inhabits, so we get the second property

(iii) Since the induction principle given by the precomposition allows us to work in any type and in any universe, the lemma 12 applies indifferently of the universe and both $\bigcirc' X$ and $\bigcirc X$ have the same universal property, hence they are equivalent.

(iv) and (v) Note that $\bigcirc(F_a)$ is an equivalence, so any $X : \mathcal{U}'$ with the property from (iv) is $F-$local. Conversely if we have an X such that it is F-local and thus $\bigcirc'$-modal, then the precomposition is an equivalence for any f such that $\bigcirc'(f)$ is an equivalence, but since the two modalities coincide in $\mathcal{U}$ we have the fourth property. Finally (v) arises from the fact that reflective subuniverses are determined by the modal types.

□

4.3 Non accessible modalities

Now that we have seen how universes work in type theory, we can introduce *propositional resizing* and with it the case of the *double negation modality*.

Propositional resizing

For any mere proposition $P : \mathbf{Prop}_{\mathcal{U}_n}$ we will also have $P : \mathbf{Prop}_{\mathcal{U}_{n+1}}$ since, as we stated, the universes are cumulative. This property gives us a natural embedding which in general is not an equivalence although it is consistent to assume that it is

Axiom 2 (Propositional resizing). The map $\mathbf{Prop}_i \to \mathbf{Prop}_{i+1}$ is an equivalence.

The axiom is called this way because it means that any *mere proposition* in $\mathcal{U}_{n+1}$ can be "resized" as an equivalent one in the universe $\mathcal{U}_n$ and it is a form of *impredicatibility* * for the mere propositions.

The double negation modality

In logic, the double negation is the operation that takes a proposition P to $\neg\neg P$, which in the classical setting is P itself. In intuitionistic logic though, this is not the case (recall that in general the *law of the excluded middle* does not hold). We are in the latter case, since in homotopy type theory the negation is

$$\neg P :\equiv (P \to \varnothing)$$

where $\varnothing$ is the empty type. Then the double negation of a proposition P is

$$\neg\neg P :\equiv (P \to \varnothing) \to \varnothing$$

which is not the same as P. As both are mere propositions we would only need to construct a map $\neg\neg P \to P$ for them to be equivalent and consequently making us able to prove by contradiction.

Definition 36. *We can define* the double negation modality *by its modal operator* $\lambda P.((P \to \varnothing) \to \varnothing)$ *and* $\eta(p) :\equiv \lambda g.g(a)$. *It forms a uniquely eliminating modality, that is,*

$$\lambda f.f \circ \lambda g.g(a) : (\Pi_{(z:\neg\neg A)} \neg\neg(P(z)) \to (\Pi_{(x:A)} \neg\neg P(\lambda g.g(x)))$$

is an equivalence for $A : \mathcal{U}$ *and* $P : \neg\neg A \to \mathcal{U}$.

*A definition is impredicative if it gives an object by pointing to a total that includes said object.

Since both the domain and codomain of this map are mere propositions, it suffices to find a map from the codomain (i.e. $(\Pi_{(x:A)}\neg\neg P(\lambda g.g(x)))$) to the domain (i.e., $\Pi_{(z:\neg\neg A)}\neg\neg(P(z))$). Consider $f : \Pi_{(x:A)}\neg\neg P(\lambda g.g(x))$ for a $P : \neg\neg A \to \mathcal{U}$.

Then given $z : \neg\neg A$ we have to derive a contradiction from $g : \neg P(z)$. Since we are proving a contradiction we can take the double negation from z and assume a term $a : A$, and since $\neg\neg A$ is a mere proposition we have $z = \lambda g.g(a)$, so we can transport $f(a)$ to get a term of $\neg\neg P(z)$, contradicting g.

Lemma 28. If propositional resizing holds, the double negation modality corresponds to the nullification at $\mathbf{2}$. Moreover, it is the propositional truncation.

A type X is $\mathbf{2}$-null if and only if X is a mere proposition. A map $\mathbf{2} \to X$ is given by two points in X. If we take X to be $\mathbf{2}$-null and $x, y : X$, then it follows that there is a unique point $z : X$ such that $x = z$ and $y = z$ and in particular it follows that $x = y$, so X is a mere proposition. It is straightforward then that $||-||$-modal types are the $\mathbf{2}$-null types. To see that those are also all the modal types for $\neg\neg$ we have to see how are the modal types of the double negation modality. Recall that the modal types are the $\neg\neg$-stable ones, i.e. , those X such that

$$\eta_X^{\neg\neg} : X \to \neg\neg X$$

is an equivalence. Since $\neg\neg X$ is already a mere proposition $\neg\neg$-modal types are a family of mere propositions. Then as we already have the $X \to \neg\neg X$ from η_X we only need to find a map $\neg\neg \to X$ to have an equivalence.

Propositional resizing implies the following version of the *Law of the excluded middle* only for the propositions

$$LEM_{-1} :\equiv \Pi_{A:\mathcal{U}} IsProp(A) \to A + \neg A$$

which is logically equivalent to the *Law of double negation*

$$\Pi_{A:\mathcal{U}} IsProp(A) \to (\neg\neg A \to A)$$

and therefore under these hypotheses the double negation is the $\mathbf{2}$-nullification.

Remark 27. in [13], Proposition 11.3, it is shown that in models for homotopy type theory based on Grothendieck $(\infty, 1)$-topoi propositional resizing as given in Axiom 2 holds. Although for example in [12] they take propositional resizing as an extra hypothesis for some constructions, this view of an ∞-topos as the model for HoTT is quite widespread.

4.4 Smallnes and non-accessible localizations

In [14] Christensen builds a localization at a family of maps whose accessibility is independent of ZFC, by reproducing the example of [15], that shows that there is a reflective subcategory of the ∞-topos of spaces with accessibility independent from ZFC. We will first construct this localization and then briefly discuss the accessibility of the object.

4.4.1 Smallness

Recall that we call a type A *small* or $\mathcal{U}$*-small* if it is a term of $\mathcal{U}$, or more precisely, if for some $B : \mathcal{U}$ there is an equivalence $A \simeq B$. For this section, we will also use the following definition involving the identity types.

Definition 37. *A type is* 0*-locally small if it is small. We define a type to be* $(n+1)$*-locally small if for any* $x, y : A$ *the type* $x = y$ *is* n*-locally small.*

Lemma 29. Let $n \geq -1$. If $f : A \to X$ is $(n-1)$-connected, A is small and X is $(n+1)$-locally small, then X is small.

Proof. For $n = -1$ we have nothing to prove since 0-locally small types are just small types. In [16], Theorem 4.6, it is shown that for A small and $A \to X$ (-1)-connected, then X is small.

Now let $x, y : X$, for a $f : A \to X$ $(n-1)$ connected, A small and X $(n+1)$-locally small. Since f is surjective we can take $x :\equiv f(a)$ and $y :\equiv f(a')$ for some $a, a' : A$. We have then $ap(f) : (a = a') \to (f(a) = f(a'))$. Note that $a = a'$ is small, $f(a) = f(a')$ n-locally small and $ap(f)$ is $(n-2)$-connected. Therefore by the induction hypothesis $f(a) = f(a')$ is small. □

Lemma 30. Let $n \geq -1$. If X is n-truncated, then X is $(n+1)$-locally small

Proof. We prove it by induction. For $n = -1$, this is just propositional resizing, which we assume during all this section. Now let $m \geq 0$ and assume that X is m-truncated. Then for $x, y : X$, the identity type $x = y$ is m-truncated. By the induction hypothesis applied to $x = y$, we see that each $x = y$ is m-locally small. That is, X is $(m+1)$-locally small, as required. □

Lemma 31. Let $\mathcal{U}$ be a universe contained in a universe $\mathcal{U}^+$. Let L be a reflective subuniverse of $\mathcal{U}^+$ such that for every $X : \mathcal{U}$, LX is small. Then the subuniverse of L-local types in $\mathcal{U}$ is reflective.

Proof. We can replace the proposition $isLocal_L : \mathcal{U}^+ \to Prop_{\mathcal{U}^+}$ that defines the subuniverse with an equivalent one landing in $Prop_{\mathcal{U}}$, using propositional resizing. So we will write $isLocal_{\bar{L}} : \mathcal{U} \to Prop_{\mathcal{U}}$ for the restriction of the new predicate to $\mathcal{U}$.

We have $is : \Pi_{X:\mathcal{U}}\Sigma_{X':\mathcal{U}}(X' \simeq LX)$. For $X : \mathcal{U}$ we will define $\bar{L}X$ to be $pr_1(is(X))$, so that $\bar{L}X : \mathcal{U}$ and $\bar{L}X \simeq LX$.

We now define

$$\eta : X \to \bar{L}X$$

as the composite $X \to LX \to \bar{L}X$, where the first map is the localization in $\mathcal{U}'$ and the second map is the inverse of the equivalence. It is easy to check that this is a localization. □

Theorem 28. *Let $n \geq -1$ and $f : \Pi_{i:I}(A_i \to B_i$ a family of $(n-1)$-connected maps, with no smallness hypotheses on I, A_i or B_i. Then the subuniverse of n-truncated and f-local types in $\mathcal{U}$ is reflective.*

Proof. First of all, note that a type is n-truncated if it is local with respect to the map $\mathbb{S}^{n+1} \to 1$. Then we will consider the family $\bar{f}$ as f together with said map and then take $\bar{f}$-local maps.

Let $\mathcal{U}^+$ be a universe containing $\mathcal{U}$, I, and all B_i and A_i. Then $\bar{f}$-local types form a reflective subuniverse in $\mathcal{U}^+$. Then we ought to show that if $X : \mathcal{U}$ then $L_{\bar{f}}X$ is small. We know by Lemma 30 that $L_{\bar{f}}X$ is $(n-1)$-locally small. Now since all the maps in $\bar{f}$ are $(n-1)$-connected, and those form the left class in an orthogonal factorization system, then $\eta : X \to L_{\bar{f}}X$ is also in the left class, and therefore $(n-1)$-connected, then by Lemma 29 $L_{\bar{f}}X$. □

4.4.2 Localization at one function

We will see now that given the right conditions, we can present this localization with a single function. This will collide with the theorem of [15], which asserts that we cannot prove the existence of such a localization from ZFC.

Axiom 3. The axiom that *sets cover* states that for every type X we have a *set* S (recall we defined sets as 0-types) and an epimorphism $S \to X$. We will say that sets cover in $\mathcal{U}$ is this is true for every $X : \mathcal{U}$.

Remark 29. The simplicial model for HoTT satisfies a form of the axiom of choice which in turn implies both that *sets cover* and *law of excluded middle for propositions*. Therefore it is not an unreasonable idea to make use of these axioms in this part of the thesis.

We can now state the main theorem of this section.

Theorem 30. *If sets cover in $\mathcal{U}$, the law of excluded middle holds and L is an accessible reflective subuniverse such that the empty type is L-local, then L can merely be presented as localization with respect to a single map.*

We will separate the statement into three propositions that we will be able to prove:

Proposition 2. Assume that sets cover in $\mathcal{U}$. Then any accessible subuniverse L of $\mathcal{U}$ can be presented as a localization with respect to a family in $\mathcal{U}$ indexed by a set.

Proof. Suppose that L is presented as a localization with respect to a family f indexed by a type $I : \mathcal{U}$. Then we can choose an epimorphism $s : S \to I$ from a set $S : \mathcal{U}$. Then L can be presented as the localization at the family $f \circ s$ indexed by S. It is easy to see that an $X : \mathcal{U}$ is f-local if and only if it is $(f \circ s)$-local, given that s is an epimorphism. □

Proposition 3. Let $f : \Pi_{i:I} A_i \to B_i$ be a family of maps. Then every f-local type is $total(f)$-local. Moreover, if I is a set with decidable equality (this is, two elements are either equal or not) and X is merely inhabited, then X is f-local if and only if it is $total(f)$-local.

Proof. Recall that for a family of maps $f : \Pi_{i:I} A_i \to B_i$, we write $total(f)$ for the induced map on Σ-types:

$$total(f) : (\Sigma_{i:I} A_i) \to (\Sigma_{i:I} B_i).$$

Then if we suppose X to be f-local, to show that it is also $total(f)$-local we must show that

$$((\Sigma_{i:I} B_i) \to X) \to (\Sigma_{i:I} A_i \to X)$$

is an equivalence. But this map is equivalent to the map

$$\Pi_{i:I} X^{B_i} \to \Pi_{i:I} X^{A_i}$$

which is an equivalence since every component is.

Now assume I to be a set with decidable equality,with X inhabited and $total(f)$-local. Then the previous map is an equivalence, and we want to show that for $i : I$ the map

$$X^{B_i} \to X^{A_i}$$

is also an equivalence. But we can make use of our assumptions to show that this last function is a retract of the previous one and thus we finish. □

Proposition 4. Let $f : \Pi_{i:I} A_i \to B_i$ be a family of maps indexed by a set I and such that the empty type is f-local. Assuming that the law of excluded middle holds for mere propositions LEM_{-1}, the f-local and $total(f)$-local types agree.

Proof. We have already shown that if X is f-local it will also be $total(f)$-local. We will prove the converse. Assume X to be $total(f)$-local, using LEM_{-1} on the propositional truncation of X, $\|X\|_{-1}$ we know that either X is merely inhabited, therefore being f-local by the previous proposition, or $\|X\|_{-1} \to \varnothing$. In this last case we have $X \to \varnothing$, therefore $X = \varnothing$ and since $\varnothing$ is f-local we have what we wanted. □

Remark 31. Note that the localization we have produced has for $n \geq 0$ the property that $\varnothing$ and in fact all propositions are $L_{\bar{f}}$-local. So if under the assumptions of the theorems we get to prove accessibility, then the localization can be presented using a single map.

4.4.3 Independence of ZFC

Theorem 6.4 in [15] reads as follows:

Theorem 32 (Theorem 6.4 of "Implications of large-cardinal principles in homotopical localization")**.** *Suppose that all cardinals are nonmeasurable. Then there is a homotopy idempotent functor E on simplicial sets that is not equivalent to f-localization for any map f.*

The localization given in the proof of such theorem can be written in type theory by choosing a family that gives the same conditions when interpreted in simplicial sets. The details of this localization are given in [1].

Now, if the localization was accessible in Homotopy Type Theory (with *sets cover* and LEM), then it could be merely presented as a localization with respect to a single map. This is what Theorem 32 shows is not provable from ZFC, assuming its consistency, and so it follows that the localization cannot be proven to be accessible.

Chapter 5

Formalization of mathematics

A formal verification of a computer program is a formalized proof that the program has specific properties, for example, to give a specified input given another input, or what sort of data the output is, or that the program will always terminate... This verification will often take form of proof that a program is of a certain type.

Thus, programming languages based on type theories are a natural place to do verification. Examples are Coq and Agda. You can write the certification at the same time as the program and run it or ignore it when running the code. There are also other program analysis tools that can produce automated proofs of certain aspects of a computer program, such as safety, and of course, these languages can be used for checking mathematics, by writing theorems as programs.

As already stated in the Introduction, one of the advantages of dependent type theory is that it is a very adequate formalization in which to base a proof assistant. Modern software like the aforementioned Coq and Agda are developed based on a Martin Löf Type Theory. In these programming languages, we cannot only write programs but also express properties of programs using types and write programs to express proofs of our programs being correct. As for why to prefer dependent type theory to set theory when doing formal verification in a computer, see this answer in MathOverflow from Andrej Bauer on the construction of proof assistants.

5.1 Agda

Agda is a dependently typed programming language, developed at Chalmers University of Technology by the programming logic group. Because of its strongly typed nature and dependent types, it can be used as a proof assistant allowing to run proofs as algorithms.

The following is an example of a simple Agda program: here we see that the program defines a *data type* called *Greeting* and a constructor *hello*. Then the function *greet* of type *Greeting* will return *Hola*.

```
data Greeting : Set where
  hello : Greeting

greet : Greeting
greet = Hola
```

Figure 5.1: Hello world program in Agda

5.2 The Unimath Library

The reasons for formalizing mathematics can vary, but the principal one is that the actual peer review process is not infallible: checking proofs can be hard and require a lot of time if you are not an expert, and having a proof assistant to verify your proofs algorithmically can be very helpful.

It can also provide new insights on the mathematical structure or the process of the proof, as it requires to build everything explicitly, and finally, it assures oneself that their arguments and computations are correct.

But when doing mathematics on a proof assistant we might find ourselves overwhelmed by the actual size of the proofs, since we start from the beginning and have to build up all the theory that precedes our theorems. We would also want our software to be reusable and modular, being able for example to change a proof and keep the rest behaving the same way. Here the use of a library plays a key role.

The Agda-Unimath project creators, Elisabeth Bonnevier, Jonathan Prieto-Cubides, and Egbert Rijke describe it as follows:

a community-driven effort aimed at formalizing mathematics from a univalent point of view using the dependently typed programming language Agda

The goal of the library is to create a sort of online encyclopedia of mathematics, formalized from the univalent point of view, so the library is organized by mathematical subjects, with one folder per subject, and the formalizations are made in *literate agda*, which combines the Agda language with markdown. This allows the treating of files as pages of a mathematics wiki.

Each file is focused on a single topic, usually giving the definition of a new concept and its basic properties or a theorem and its immediate corollaries. For instance, this is the view in the Agda-Unimath page for the file regarding the 0-modality:

The zero modality

```
module orthogonal-factorization-systems.zero-modality where
```

▶ Imports

Idea

The **zero modality** is the modality that maps every type to the unit type.

Definition

```
operator-zero-modality :
  (l1 l2 : Level) → operator-modality l1 l2
operator-zero-modality l1 l2 _ = raise-unit l2

unit-zero-modality :
  {l1 l2 : Level} → unit-modality (operator-zero-modality l1 l2)
unit-zero-modality _ = raise-star
```

Properties

The zero modality is a uniquely eliminating modality

```
is-uniquely-eliminating-modality-zero-modality :
  {l1 l2 : Level} →
  is-uniquely-eliminating-modality (unit-zero-modality {l1} {l2})
is-uniquely-eliminating-modality-zero-modality {l2 = l2} A P =
  is-local-is-contr
    ( unit-zero-modality)
    ( raise-unit l2)
    ( is-contr-raise-unit)
```

The zero modality is equivalent to –2-truncation

This remains to be made formal.

Figure 5.2: Screenshot from the agda-unimath webpage

5.3 Modalities in the Agda Unimath Library

Most of the papers cited as references for the theory of localizations and modalities [1; 2; 14] have been formalized by their authors either in their own repositories or in the Coq-Hott library. At the start of this work none of it was formalized in Agda.

Since then there has been an effort to formalize the content of [2] in the Agda-Unimath library, by one of its maintainers and significative advance has been made. In the folder *Orthogonal factorization systems* one can find 33 different agda files, containing from examples of higher modalities to the different definitions of a modality.

The author of the thesis has also started formalizing part of it, sometimes overlapping the already formalized topics and sometimes building on them. The objective of this exercise is to formalize the results on accessibility and in a future to be able construct the example from [15] in a type-theoretic language. The progress will be posted in the personal repository of the author.

5.4 Conclusions

We have seen some very concrete examples of non-accessible modalities and localizations, but the main question is still open: *under what conditions can we say that modalities correspond to nullifications?* Our examples direct us to universes and resizing principles, such as the propositional resizing axiom, as we have seen that the double negation becomes accessible if propositional resizing is assumed. We have also proved that we can define modalities in higher universes that "fit" an existing modality if this last is accessible.

The way of producing a non-accessible modality in [1], using the construction from [15] and showing that it cannot be proven to be accessible, is not ideal since we are making these kind of jumps between theories. A way to follow down this path might be to construct this family in the HoTT language and assuming that it is accessible derive something that we expect not to be provable. This would be only fully in homotopy type theory so it would avoid any subtlety in the change of logics.

Finally on a more ambitious note, in [15] it is shown that *Vopenka's principle* implies that all localizations are accessible. The study of this behaviour in category theory and the study of large cardinals and *how* they alter set theory could shed light on the topic in homotopy type theory. Since types are not "collections of terms" we do not have an intuitive form of cardinals, but in dependent type theory one can define finite types and in [12] the type of cardinal numbers is defined as

$$Card :\equiv \|Set\|_0,$$

the 0-truncation of the type *Set* of sets, and a cumulative hierarchy in a universe as a higher inductive type. So maybe some similar principles could be derived.

References

[1] J. D. Christensen, M. Opie, E. Rijke, and L. Scoccola, "Localization in homotopy type theory," 2020.

[2] E. Rijke, M. Shulman, and B. Spitters, "Modalities in homotopy type theory," *Logical Methods in Computer Science*, vol. Volume 16, Issue 1, Jan. 2020.

[3] A. Bousfield, "The localization of spaces with respect to homology," *Topology*, vol. 14, no. 2, pp. 133–150, 1975.

[4] D. P. Sullivan and A. Ranicki, "Geometric topology: Localization, periodicity and galois symmetry: The 1970 mit notes," 2005.

[5] J. Lurie, *Higher Topos Theory*. 2008.

[6] M. Hofmann and T. Streicher, "The groupoid interpretation of type theory," vol. 36, pp. 83–111, 1998.

[7] S. Awodey and M. A. Warren, "Homotopy theoretic models of identity types," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 146, no. 1, p. 45–55, 2009.

[8] V. Voevodsky, "A very short note on homotopy λ-calculus," 2006.

[9] S. Awodey and M. Warren, *Homotopy Type Theory*. Cambridge: Cambridge University Press, 2013.

[10] E. Rijke, "Introduction to homotopy type theory," 2022.

[11] P. Martin-Löf, *Intuitionistic type theory*, vol. 1 of *Studies in proof theory*. Bibliopolis, 1984.

[12] V. Voevodsky, S. Awodey, and D. Licata, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Princeton, NJ: Institute for Advanced Study, 2013.

[13] M. Shulman, "All $(\infty, 1)$-toposes have strict univalent universes," 2019.

[14] J. D. Christensen, "Non-accessible localizations," 2021.

[15] C. Casacuberta, D. Scevenels, and J. H. Smith, "Implications of large-cardinal principles in homotopical localization," *Advances in Mathematics*, vol. 197, no. 1, pp. 120–139, 2005.

[16] E. Rijke, "The join construction," 2017.

[17] F. Cherubini and E. Rijke, "Modal descent," *Mathematical Structures in Computer Science*, vol. 31, pp. 363–391, oct 2020.

[18] M. Anel, G. Biedermann, E. Finster, and A. Joyal, "Left-exact localizations of ∞-topoi ii: Grothendieck topologies," 2023.

[19] M. Anel, G. Biedermann, E. Finster, and A. Joyal, "Left-exact localizations of ∞-topoi i: Higher sheaves," *Advances in Mathematics*, vol. 400, p. 108268, may 2022.

[20] T. Lawson, "An introduction to bousfield localization," 2020.

[21] K. Buzzard, "The Xena Project Blog." https://xenaproject.wordpress.com, 2023. [Online; accessed 27-August-2023].

[22] nLab authors, "Hedberg's theorem." https://ncatlab.org/nlab/show/Hedberg%27s+theorem, Aug. 2023. Revision 1.

[23] Giovanni Sambin, "Intuitionistic type theory, per martin löf, notes by giovanni sambin of a series of lectures given in padua, june 1980." https://archive-pml.github.io/martin-lof/pdfs/Bibliopolis-Book-retypeset-1984.pdf, 1980.

[24] "Agda documentation." https://agda.readthedocs.io/en/v2.6.3/, 2023. .

[25] E. Rijke, E. Bonnevier, J. Prieto-Cubides, F. Bakke, and others, "The agda-unimath library."