# Lawvere-Tierney Sheafification in Homotopy Type Theory

By

Kevin Quirin

Major Subject: Computer Science

Approved by the
Examining Committee:

Pierre Cointe, Thesis Advisor

Nicolas Tabareau, Thesis Advisor

Carl F. Gauß, Göttingen

Euclid, Athens

Mines de Nantes

2016

### Abstract

Write the abstract

# Acknowlegdments

Thanks to.

Write the acknowledgments

# Contents

**Acknowlegdments** **iii**

**Contents** **iv**

**1 Introduction** **1**

**2 Homotopy type theory** **3**
2.1 Dependent type theory . . . . . . . . . . . . . . . . . . . . . . . . 3
2.2 Identity types and Univalence axiom . . . . . . . . . . . . . . . . 7
2.3 Higher Inductives Types . . . . . . . . . . . . . . . . . . . . . . . 10
2.4 Introduction to homotopy type theory . . . . . . . . . . . . . . 13

**3 Higher modalities** **19**
3.1 Modalities . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.2 Examples of modalities . . . . . . . . . . . . . . . . . . . . . . . 20
3.3 New type theories . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.4 Truncated modalities . . . . . . . . . . . . . . . . . . . . . . . . 20
3.5 Translation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

**4 Colimits** **21**
4.1 Colimits over graphs . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.2 Van Doorn's and Boulier's constructions . . . . . . . . . . . . . 23
4.3 Towards groupoid objects . . . . . . . . . . . . . . . . . . . . . . 23

**5 Sheaves in homotopy type theory** **25**
5.1 Sheaves in topoi . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
5.2 Sheaves in homotopy type theory . . . . . . . . . . . . . . . . . . 27
5.3 Forcing in type theory . . . . . . . . . . . . . . . . . . . . . . . . 41

**6 Conclusion and future works** **43**

**List of Notations** **47**

**Bibliography** **49**

# 1 Introduction

Here is the intro.

Write the intro

# 2 Homotopy type theory

Write an intro to chapter HoTT

## 2.1 Dependent type theory

In Zermelo-Frankæl set theory, the most basic assertion is

$$x \in E$$

where $x$ and $E$ are sets. In dependent type theory, a similar judgement can be

$$a : A,$$

to be read as "$a$ is of type $A$". The main difference with membership relation is that an element $a$ has one and only one type, while we can say $x \in E$ and $x \in F$ for the same element $x$ in set theory (it is the definition of $x \in E \cap F$).

Dependent (or Martin-Lóf, due to its inventor [Mar98]) type theory is based on the Curry-Howard correspondance [How80], or propositions-as-types principle. Indeed, we do not need to make a difference between types an propositions. Hence $a : A$ will be read "$a$ is of type $A$" when $A$ is seen as a type, and "$a$ is a proof of $A$" when $A$ is seen as a proposition. In the rest of this section 2.1, we present the different types used to build dependent type theory. It is not intended to be complete ; for example we sometimes only give the non-dependent elimination rules to ease the reading. The reader can refer to [NPS01] or [HoTT] for a more detailed introduction.

### Universes

In dependent type theory, types are also terms of a universe Type. Of course, we want the universe Type to be itself a type, and the Russel's paradox is close here. We solve the problem by using a cumulative hierarchy of universe

$$\mathrm{Type}^0 : \mathrm{Type}^1 : \mathrm{Type}^2 : \cdots$$

every universe $\mathrm{Type}^i$ is of type $\mathrm{Type}^{i+1}$. Cumulativity means that if $A : \mathrm{Type}^k$ and $k < m$, then $A : \mathrm{Type}^m$. The handling of universes level can be done automatically by proof assistants such as Coq, and we will thus use only a "type of type" Type, to be understood in a polymorphic way.

### Empty and Unit types

The first two types we will see are the Empty type (denoted $\mathbf{0}$) and the Unit type (denoted $\mathbf{1}$). These are respectively the types with zero and one elements (named $\star$). Those two types are dual to each other:

- having a term of type $\mathbf{0}$ in the context allows to prove anything, while having a term of type $\mathbf{1}$ in the context is useless
- dually, giving a term of type $\mathbf{1}$ is trivial, while giving a term of type $\mathbf{0}$ is impossible (if the theory if consistent).

Here are the (non-trivial) introduction and elimition rules for these types:

$$\frac{\Gamma \vdash x : \mathbf{0} \qquad X : \mathrm{Type}}{\Gamma \vdash X}\ \mathbf{0}\text{-}\textsc{elim} \qquad\qquad \frac{}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-}\textsc{intro}$$

Under the propositions-as-types principle, $\mathbf{0}$ is the type always false, and $\mathbf{1}$ the type always true. With a categorical point of view, $\mathbf{0}$ is an initial object and $\mathbf{1}$ is a terminal object.

### Coproduct

The coproduct of $A$ and $B$, noted $A+B$, is seen as the disjoint sum of $A$ and $B$. It is described by the inductive type generated by

$$\left|\begin{array}{lcl} \mathrm{inl} & : & A \to A+B \\ \mathrm{inr} & : & B \to A+B \end{array}\right.$$

The introduction and elimination rules of coproduct are:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{inl}\, a : A+B}\ +\text{-}\textsc{intro}_L \qquad\qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathrm{inr}\, b : A+B}\ +\text{-}\textsc{intro}_R$$

$$\frac{\Gamma \vdash p : A+B \qquad \Gamma, x : A \vdash c_A : C \qquad \Gamma, x : B \vdash c_B : C}{\Gamma, \vdash \mathrm{sum\_rect}(p, c_A, c_B) : C}\ +\text{-}\textsc{elim}$$

Under the propositions-as-types principle, $A+B$ is seen as the disjunction of $A$ and $B$.

### Dependent product

One of the things both mathematicians and computer scientists love to do is to define functions. If $A$ and $B$ are types, one can consider the type of functions from $A$ to $B$, taking an inhabitant of type $A$ (called the source type) and giving an inhabitant of type $B$ (the target type). What is new in dependent type theory is that the target type is allowed to depend on the argument of the function.
Example – For example, one can consider a function taking a natural number $n$ and giving a natural number greater than $n$. The target source depends indeed of $n$.

The type of dependent functions with source type $A$ and target type $Bx$ is called "dependent product over $B$" (or "pi-type over $B$"), and will be denoted $\prod_{x:A} Bx$ or $(x:A) \to (Ba)$. When $B$ does not depend on $A$, we just say "arrow type" and note it $A \to B$.

The introduction and elimination rules of dependent products are:

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda(x:A), b : \prod_{x:A} B}\ \textstyle\prod\text{-}\textsc{intro} \qquad \frac{\Gamma \vdash f : \prod_{x:A} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \textstyle\prod\text{-}\textsc{elim}$$

Under the propositions-as-types, type of non-dependent functions $A \to B$ is seen as implication $A \Rightarrow B$, and type of dependent functions $\prod_{x:A} Bx$ is seen as universally quantified formulas $\forall x, Bx$.

### Dependent sum

If $A$ and $B$ are two types, we would like to define the type of pairs $(a,b)$, where $a : A$ and $b : B$. The resulting type is called the product of $A$ and $B$, noted $A \times B$.

As for functions, dependent type theory allows the second type to depend on the first type. Thus, the type of pairs where the first element $x$ is in type $A$ and the second element $y$ is in type $Bx$ is called "dependent sum over $B$" (or sigma-type over $B$), noted $\sum_{a:A} Ba$.

The introduction and elimination rules are:

$$\frac{\Gamma, x : A \vdash B : \text{Type} \qquad \Gamma \vdash x : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : \sum_{x:a} B}\ \textstyle\sum\text{-}\textsc{intro}$$

$$\frac{\Gamma \vdash p : \sum_{x:A} B}{\Gamma \vdash \pi_1 p : A}\ \textstyle\sum\text{-}\textsc{intro}_1 \qquad \frac{\Gamma \vdash p : \sum_{x:A} B}{\Gamma \vdash \pi_2 p : B[\pi_1 p/x]}\ \textstyle\sum\text{-}\textsc{intro}_2$$

The projections $\pi_1$ and $\pi_2$ will sometimes be noted, when applied to a term $u$, $u_1$ and $u_2$.
Nota – We note that the terminology might be confusing: the dependent generalization of products are dependent sums, while dependent products are generalization of functions.

Under the propositions-as-types principle, $A \times B$ is seen as the conjonction $A \wedge B$, and $\sum_{x:A} Ba$ is seen as the existentially quantified formula $\exists x Bx$.

### Inductive types

Dependent type theory actually allows us to define any inductive type. An inductive type is a type defined only by its introduction rules, in a free way. Its elimination rules are automatically determined. We have already seen examples of inductive types: Empty, Unit, Coproduct.

The most basic example of inductive types might be the type $\mathbb{N}$ of naturals. Its introduction rules are

$$\frac{}{0 : \mathbb{N}}\ \mathbb{N}\text{-}\textsc{intro}_0 \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash S\,n : \mathbb{N}}\ \mathbb{N}\text{-}\textsc{intro}_S$$

We also say that its *constructors* are $0 : \mathbb{N}$ and $S : \mathbb{N} \to \mathbb{N}$: $\mathbb{N}$ is thus the free monoid generated by 0 and $S$. The elimination rule for $\mathbb{N}$ is the famous induction principle

$$\dfrac{\begin{array}{cc} \Gamma, x : \mathbb{N} \vdash P : \mathrm{Type} & \Gamma \vdash c_0 : C[0/x] \\ \Gamma \vdash n : \mathbb{N} & \Gamma, n : \mathbb{N}, y : C \vdash c_S : C[S\,n/x] \end{array}}{\Gamma \vdash \mathrm{nat\_ind}(C, c_0, c_S(n,y), n) : C[n/x]}\ \mathbb{N}\text{-}\textsc{elim}$$

This elimination rule allows us to define basic operators on natural numbers: addition, multiplication, order, *etc.*

### Paths type

One of the most powerful tool in dependent type theory might be the identity types. They allow us to talk about propositional equality between inhabitants of a type. The identity type over $A$ will be noted $a =_A b$ or $a = b$ if $A$ can be inferred from context . What is great about identity type over $A$ is that its definition does not depend on the type $A$. If $a, b : A$, $a =_A b$ is defined as the inductive type whose only constructor is

$$\mathrm{idpath} : \prod_{a:A} a =_A a$$

($\mathrm{idpath}_x$ will sometimes be just noted $1_x$, or even 1).

$$\dfrac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{idpath}_a : a =_A a}\ =\text{-}\textsc{intro}$$

$$\dfrac{\begin{array}{cc} \Gamma \vdash a, b : A & \Gamma, x : A, y : A, p : x =_A y \vdash P : \mathrm{Type} \\ \Gamma \vdash q : a =_A b & \Gamma, z : A \vdash w : P[z, z, 1/x, y, p] \end{array}}{\Gamma \vdash \mathrm{path\_ind}_P^q(w) : P[a, b, q/x, y, p]}\ =\text{-}\textsc{elim}$$

The elimination principle should be looked closer. Lets simplify to understand how it works: suppose that the type family $P$ depends only on two objects $x, y : A$. Then if $C$ is reflexive, *i.e* if $C(x,x)$ is always inhabited, then whenever $x = y$, $C(x,y)$ is inhabited too.

Path-induction, also known as "$J$ principle", allow us to state the following:

**Lemma 1 : [HoTT, Lemma 2.3.1]**
*Let* $A : \mathrm{Type}$ *and* $P : A \to \mathrm{Type}$ *a type family over* $A$*. Then if* $p : x =_A y$*, there is a function* $\mathrm{transport}_P^p : P\,x \to P\,y$.

*Proof.* Let's do this proof to use path-induction for the first time. Let $Q$ be the type family indexed by $x, y : A$

$$Q(x,y) := P\,x \to P\,y.$$

Then by path-induction, it suffices to define $Q(x,x)$. The latter is clearly inhabited by $\mathrm{idmap}_{P\,x}$. □

Path-induction might be the most powerful tool we can use. For example, most of the lemmas we will see in section 2.4 can be proved using path-induction.

### Summary

We can summarize the situation in the following array:

| Name | Notation | Proposition-as-types |
|---|---|---|
| Empty | $\mathbf{0}$ | $\perp$ |
| Unit | $\mathbf{1}$ | $\top$ |
| Coproduct, sum | $A+B$ | $A \vee B$ |
| Function | $A \times B$ | $A \Rightarrow B$ |
| Dependent function, pi-type | $\prod_{x:A} B\,x$ | $\forall x, B\,x$ |
| Product | $A \times B$ | $A \wedge B$ |
| Dependent sum, sigma-type | $\sum_{x:A} B\,x$ | $\exists x, B\,x$ |

## 2.2 Identity types and Univalence axiom

Identity types are very useful to assert propositional equalities between object. Note that it does not characterize *judgemental* equality, which we consider here as belonging to the meta-theory (some theories, as Voevodsky's Homotopy Type System [HTS], implemented in proof assistant Andromeda [m31]). One can prove that identity types give each type a structure of a $\omega$-groupoid, *i.e* the structure of a $\omega$-category where all arrows are invertible. For example, identity types over a type $A$ satisfies:

- Reflexivity: for all $x : A$,
$$1_x : x = x$$
- Transitivity: for all $x,y,z : A$, $p : x = y$ and $q : y = z$,
$$p \cdot q : x = z$$
- Symmetry: for all $x,y : A$ and $p : x = y$, $p^{-1} : y = x$

- Reflexivity and symmetry behave well together: for all $x,y : A$ and $p : x = y$,

  $$p \cdot p^{-1} = 1_x \text{ and } p^{-1} \cdot p = 1_y$$

  Note that these paths are respectively in types $x = x$ and $y = y$.

- Reflexivity and associativity behave well together: for all $x,y : A$ and $p : x = y$,

  $$p \cdot 1 = p \text{ and } 1 \cdot p = p$$

- Associativity: for all $w,x,y,z : A$ and $p : w = x$, $q : x = y$, $r : y = z$,

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

- Maclane pentagon: for all $v,w,x,y,z : A$, $p : v = w$, $q : w = x$, $r : x = y$ and $s : y = z$, the following diagram involving associativities commutes

$$\begin{tikzcd}
 & p \cdot (q \cdot (r \cdot s)) \arrow[dl] \arrow[dr] & \\
(p \cdot q) \cdot (r \cdot s) \arrow[d] & & p \cdot ((q \cdot r) \cdot s) \arrow[d] \\
((p \cdot q) \cdot r) \cdot s \arrow[rr] & & (p \cdot (q \cdot r)) \cdot s
\end{tikzcd}$$

- The coherences goes on and on.

A thing one can notice here is that we work in a proof-relevant setting. The types $x = y$ are just types, and thus can be inhabited by different objects. It might sound unfamiliar for mathematicians, but we will give in section 2.4 an interpretation of identity types justifying proof-relevance.

Identity types also have a good behavior with respect to function. Let $A, B$ : Type and $f : A \to B$ ; then

- For all $x,y : A$, there is a map $\mathrm{ap}_f : x = y \to f(x) = f(y)$
- It is coherent with inverse: $\mathrm{ap}_f p^{-1} = (\mathrm{ap}_f p)^{-1}$
- It actually is coherent with the whole $\omega$-groupoid structure. We refer to [HoTT] for more detailed results.

The fact that the definition of identity types does not depend on the type might seem strange, but it actually allows to catch the equality one would want ; for example, we can characterize the equalities between dependent pairs:

**Lemma 2 : Paths in dependent sums**
*Let $A$ : Type and $B : A \to$ Type. Then, for any $u, v : \sum_{x:A} B\,a$, we have a term*

$$\mathrm{path}_{\Sigma} : \sum_{p:u_1 =_A v_1} \mathrm{transport}^{p}_{p}\, u_2 = v_2 \to u = v.$$

This characterization goes even further, as we can prove that $\mathrm{path}_{\Sigma}$ is an equivalence. Equivalences are very important objects in homotopy type theory, and are defined as

**Definition 3 : Equivalence**
*Let $A, B$ : Type and $f : A \to B$. Wa say that $f$ is an equivalence, noted $\mathrm{IsEquiv}(f)$, if there are:*

- *a map $g : B \to A$, called the inverse of $f$*
- *a term $\mathrm{retr}_f : \prod_{x:A} g(f(x)) = x$, called the retraction of the equivalence*
- *a term $\mathrm{sect}_f : \prod_{x:A} f(g(x)) = x$, called the section of the equivalence*
- *a term $\mathrm{adj}_f : \prod_{x:A} \mathrm{ap}_f \mathrm{retr}_f x = \mathrm{sect}_f (f\,x)$, called the adjunction of the equivalence*

*We say that $A$ and $B$ are equivalent, noted $A \simeq B$, if there is a function $f : A \to B$ which is an equivalence.*

Hence, $\mathrm{path}_{\Sigma}$ allows us to prove equalities in a dependent sum $\sum_{x:A} B\,a$, but its inverse allows us to prove, from an equality in the dependent sum, equalities in $A$ and in $B\,a$.

Unfortunately, identity types does not allow to characterize equalities for all types. One example is paths in dependent products. In usual mathematics, one would like pointwise equality to be a sufficient (actually, a necessary and sufficient) property to state equality between function. In type theory, that would be phrased

$$\prod_{x:A} f\,x = g\,x \to f = g. \tag{2.1}$$

But one can prove (see [Str93] for a semantical proof) that this property, called *functional extensionality* cannot be proved from rules of dependent type theory. All we can say is that the backward function of 2.1 can be defined, called happly:

$$\mathrm{happly} : f = g \to \prod_{x:A} f\,x = g\,x.$$

Functional extensionality will thus be stated as the axiom:

**Axiom 4 : Functional extensionality**
*For any* $A : \mathrm{Type}$ *and* $B : A \to \mathrm{Type}$*, the arrow* happly *is an equivalence.*

In a way, that could solve our problem. But dependent products are not the only types for which we cannot characterize identity types; the same problem arises with identity types of the universe. An answer was proposed first by Martin Hofmann and Thomas Streicher in 1996 in [HS96], and later (around 2005) by Vladimir Voevodsky: the univalence axiom. As for functional extensionality, it asserts that a certain arrow is an equivalence.

**Axiom 5 : Univalence axiom**
*For any types* $A, B : \mathrm{Type}$*, the function*

$$\mathrm{idtoequiv} : A = B \to A \simeq B$$

*is an equivalence.*

What is great about univalence axiom is that is seems to imply all other characterization of identity types. For example:

**Lemma 6 : [BL11][Lic]**
*Univalence axiom implies functional extensionality.*

Another example the reader can think of is the type of streams (and more generaly coindutive) extensionality: we want two streams (infinite list) to be equal when they are pointwise equal. It seems that univalence axiom implies it [Lic].

The main issue with univalence is that it clearly oblige us to work in a proof-relevant setting ; indeed, there are for examples two distinct proofs of $\mathbf{1}+\mathbf{1} = \mathbf{1}+\mathbf{1}$: the identity, and the function swapping left and right.

## 2.3 Higher Inductives Types

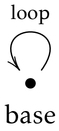
As seen in section 2.1, one can define a type by giving only its constructors ; but we also saw in section 2.2 that our setting is proof-relevant. It means that a type is characterized not only by its objects, but also by its identity types between these objects, and identity types of these identity types, *etc.* Hence, we would like to be able, like for inductive types, to define a type by giving constructors for the objects, and constructors for the identity types, *etc.* Then, the constructed type is the type freely generated by the constructors, and whose identity types are freely generated by the constructors, *etc.* Unlike inductive types, it is not known if elimination rules of higher inductive type can easily (and automatically) be infered from the constructors. At the moment, we prefer to express them explicitly. As a general theory of hogher inductive types would be very hard, we will only give some fundamental examples.

### The circle

The most basic example of higher inductive type is the circle $\mathbb{S}^1$. The circle consists of just a point, with a non-trivial path above this point. It is defined as the higher inductive type generated by

$$\left|\begin{array}{lcl} \mathrm{base} & : & \mathbb{S}^1 \\ \mathrm{loop} & : & \mathrm{base} = \mathrm{base} \end{array}\right.$$

It can be pictured as



As said, we give the elimination principle of the circle. We start by the non-dependent eliminator.

**Lemma 7 : Non-dependent elimination of $\mathbb{S}^1$**
*Let $P$ be a type. If $b : P$ and $p : b = b$, then there is a map*

$$\mathbb{S}^1_{\mathrm{rec}} : \mathbb{S}^1 \to P$$

*such that $\mathbb{S}^1_{\mathrm{rec}}(\mathrm{base}) \equiv b$ (judgementally) and $\mathrm{ap}_{\mathbb{S}^1_{\mathrm{rec}}}(\mathrm{loop}) = p$ (propositionally).*

It says that if in $P$ you can find a "copy" of $\mathbb{S}^1$, then there is a "good" function $\mathbb{S}^1 \to P$. The dependent eliminator is a bit more complicated, but still understandable.

**Lemma 8 : Dependent elimination of $\mathbb{S}^1$**
*Let $P : \mathbb{S}^1 \to \mathrm{Type}$ be a type family over $\mathbb{S}^1$. If $b : P(\mathrm{base})$ and $p : \mathrm{transport}_P^{\mathrm{loop}}(b) = b$, then there is a term*

$$\mathbb{S}^1_{\mathrm{ind}} : \prod_{x:\mathbb{S}^1} P\,x$$

*such that $\mathbb{S}^1_{\mathrm{ind}}(base) \equiv b$ and $\mathrm{ap}_{\mathbb{S}^1_{\mathrm{ind}}}(\mathrm{loop}) = p$.*

The most famous result about the circle is the computation of its homotopy group $\pi_1(\mathbb{S}^1)$.

**Proposition 9 : Fundamental group of $\mathbb{S}^1$ [LS13]**
*There is an equivalence*

$$(\mathrm{base} = \mathrm{base}) \simeq \mathbb{Z}$$

*where* $\mathbb{Z} \overset{def}{=} \mathbb{N} + \mathbb{N} + \mathbf{1}$.

### Coequalizers

Another example of higher inductive type is the computation of coequalizers. In category theory, a coequalizer of two objects $a$ and $b$ with arrow $f, g \in \mathrm{Hom}(a,b)$ is a $c \in \mathrm{Obj}$ together with an arrow $q \in \mathrm{Hom}(b,c)$ such that $\begin{tikzcd} a \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & b \arrow[r, "q"] & c \end{tikzcd}$ commutes, and which is universal with respect to this property, in the sense that for any other object $c' \in \mathrm{Obj}$ with $q' \in \mathrm{Hom}(b,c')$, there is an unique arrow $d : \mathrm{Hom}(c,c')$. It might be seen as the diagram

$$\begin{tikzcd}
a \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & b \arrow[r, "q"] \arrow[dr, "q'"'] & c \arrow[d, dotted, "d"', "!"] \\
& & c'
\end{tikzcd}$$

In homotopy type theory, the coequalizer of two functions $f, g : A \to B$ is defined as the higher inductive type $\mathrm{Coeq}^{f,g}$ generated by

$$\left|\begin{array}{lcl} q & : & B \to \mathrm{Coeq}^{f,g} \\ \alpha & : & \prod_{x:A} q \circ f(x) = q \circ g(x) \end{array}\right.$$

Its elimination principles are:

**Lemma 10 : Elimination of coequalizer**
*Let $A, B, f, g$ be as above.*

- *Let $P : Type$. If there are terms $q' : B \to P$ and $\alpha' : \prod_{x:A} q' \circ f(x) = q' \circ g(x)$, then there is a map*
$$\mathrm{Coeq}^{f,g}_{\mathrm{rec}} : \mathrm{Coeq}^{f,g} \to P$$
*such that for all $b : B$, $\mathrm{Coeq}^{f,g}_{\mathrm{rec}}(q\,b) \equiv q'\,b$ and $\mathrm{ap}_{\mathrm{Coeq}^{f,g}_{\mathrm{rec}}}(\alpha\,b) = \alpha'\,b$.*

- *Let $P : \mathrm{Coeq}^{f,g} \to \mathrm{Type}$. If there are terms $q' : \prod_{b:B} P(q\,b)$ and $\alpha' : \prod_{b:B} \mathrm{transport}^{\alpha\,b}_{P}(q' \circ f(b)) = q' \circ g(b)$, then there is a dependent map*
$$\mathrm{Coeq}^{f,g}_{\mathrm{ind}} : \prod_{x:\mathrm{Coeq}^{f,g}} P\,x$$
*such that for all $b : B$, $\mathrm{Coeq}^{f,g}_{\mathrm{ind}}(q\,b) \equiv q'\,b$ and $\mathrm{ap}_{\mathrm{Coeq}^{f,g}_{\mathrm{ind}}}(\alpha\,b) = \alpha'\,b$.*

It satisfies the desired universal property in the following sense:

- If $Q$ is a type and $\varphi : \mathrm{Coeq}^{f,g} \to Q$, then one can define a map $\psi : B \to Q$ such that for all $b : B$, $q \circ f(b) = q \circ g(b)$.

- One can prove that the map $\phi \mapsto \psi$ is an equivalence: from any other type such that the diagram commutes, there is an unique arrow $\mathrm{Coeq}^{f,g} \to Q$.

Some results about coequalizers in homotopy type theory can be found in the form of Coq code in the HoTT/Coq library [HoTT/Coq].

### Suspension

There is a way to see any type $A$ as the identity type of another type: the suspension of $A$, noted $\Sigma A$. It is the higher inductive type generated by

$$\left|\begin{array}{lcl} N & : & \Sigma A \\ S & : & \Sigma A \\ \mathrm{merid} & : & A \to (N = S) \end{array}\right.$$

Its elimination principle are:

**Lemma 11 : Elimination of suspension**
*Let $A$ be a type.*

- *Let $P$ : Type. If there are $n, s : B$ and $m : A \to (n = s)$, then there is a map*

  $$\Sigma^A_{\mathrm{rec}} : \Sigma A \to B$$

  *such that $\Sigma^A_{\mathrm{rec}}(N) \equiv n$, $\Sigma^A_{\mathrm{rec}}(S) \equiv s$ and for all $a : A$, $\mathrm{ap}_{\Sigma^A_{\mathrm{rec}}}(\mathrm{merid}(a)) = m(a)$*

- *Let $P : \Sigma A \to$ Type. If there are $n : P(N)$, $s : P(S)$ and $m : \prod_{a:A} \mathrm{transport}_P^{\mathrm{merid}(a)}\, n = s$, then there is a dependent map*

  $$\Sigma^A_{\mathrm{ind}} : \prod_{a:A} P\, a$$

  *such that $\Sigma^A_{\mathrm{ind}}(S) \equiv s$, $\Sigma^A_{\mathrm{ind}}(N) \equiv n$ and $\mathrm{ap}_{\Sigma^A_{\mathrm{ind}}}(\mathrm{merid}(a)) = m(a)$.*

The first thing we can notice is that $\Sigma(\mathbf{1}+\mathbf{1}) \simeq \mathbb{S}^1$ [HoTT, Lemma 6.5.1]. Actually, we can define the sequence of $n$-spheres inductively by

$$\begin{array}{ll} \mathbb{S}^0 & \overset{def}{=} \mathbf{1}+\mathbf{1} \\ \mathbb{S}^{n+1} & \overset{def}{=} \Sigma\mathbb{S}^n \end{array}$$

## 2.4 Introduction to homotopy type theory

It is now time to really introduce homotopy type theory: it is the sum of sections 2.1, 2.2 and 2.3. In other words, homotopy type theory is just dependent type theory, augmented with univalence axiom and higher inductive types. We promised in section 2.2 an interpretation of identity types. In homotopy type theory, we view types as topological spaces and inhabitants of types as points of these spaces. Then, if $x,y : A$, the identity type $x = y$ is viewed as the topological space of paths (or continuous maps $f : [0,1] \to A$ such that $f(0) = x$ and $f(1) = y$) between points $x$ and $y$ in $A$. As said, $x = y$ is itself a topological space, thus we can iterate the analogy. A path $r : p = q$ where $p,q : x = y$ are themselves paths is called a *homotopy* ; a path between paths between paths is called a 2-*homotopy, etc.* The situation can be depicted as in figure 2.1. Indeed, that interpretation allows to explain all properties of identity types.

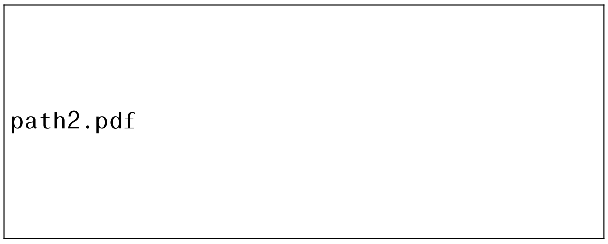


Figure 2.1: Paths: $a$ and $b$ are points, $p_1$ and $p_2$ paths, $H_1$ and $H_2$ homotopies.

- The equality $1_x$ is just the constant path

$$1_x : \begin{array}{ccc} [0,1] & \longrightarrow & A \\ t & \longmapsto & x \end{array}$$

- The inverse $\bullet^{-1}$ is the function changing a path $f$ into

$$g : \begin{array}{ccc} [0,1] & \longrightarrow & A \\ t & \longmapsto & f(1-t) \end{array}$$

- The concatenation $\bullet \cdot \bullet$ is the function changing paths $f$ and $g$ into

$$h : \begin{array}{ccc} [0,1] & \longrightarrow & A \\ t & \longmapsto & \begin{cases} f(2t) & \text{if } t \in [0, 1/2] \\ g(2t-1) & \text{if } t \in [1/2, 1] \end{cases} \end{array}$$

- A path between paths $f$ and $g$ between points $x$ and $y$ is a continuous function

$$H : \begin{array}{ccc} [0,1] & \longrightarrow & A \\ (u,v) & \longmapsto & H(u,v) \end{array}$$

  such that $H(t,0) = f(t)$, $H(t,1) = g(t)$, $H(0,s) = x$ and $H(1,s) = y$. A homotopy is then a continuous deformation of $f$ into $g$, fixing the ending points $x$ and $y$.

The table 2.1 summarize the three points of view (type theoretic, groupoidal, homotopy theoretic). In the rest of the thesis, we will use any of the following

| Type theory | Homotopy theory | $\omega$-groupoid |
|---|---|---|
| Type | Topological space | $\omega$-groupoid |
| Inhabitant | Point | Object |
| Equality | Path | Morphism |
| $\mathrm{idpath}_x$ | Constant path | Identity morphism |
| $\bullet^{-1}$ | Inverse path | Inverse morphism |
| $\bullet \cdot \bullet$ | Concatenation of paths | Composition of morphisms |
| Equality between equalities | Homotopy | 2-morphism |

Table 2.1: Three points of view about HoTT

name (*e.g* we will talk about "points of a type", "path betwenn inhabitants", *etc.*).

What homotopy theorists love to do is to compute fundamental groups (*i.e* the group of paths between two points) of different spaces ([WX10][Hut11]). We can then categorized spaces with the level at which their homotopy groups become trivial. Voevodsky has realized that this notion admits a compact inductive definition internal to type theory, given by

These references were added without even opening them. Maybe a check would be good

**Definition 12 : Truncated types**
Is-$n$-type *is defined by induction on* $n \geqslant -2$*:*

- Is-$(-2)$-type$(X)$ *if* $X$ *is a contractible type,* i.e $X$ *is inhabited by* $c : X$*, and every other point in* $X$ *is connected to* $c$*.*
- Is-$(n+1)$-type$(X) \overset{def}{=} \prod_{x,y:X}$ Is-$n$-type$(x = y)$*.*

*Then,* $\mathrm{Type}_n \overset{def}{=} \sum_{X:\mathrm{Type}}$ Is-$n$-type$(X)$*.*

For the first values of $n$, there are different names: $(-2)$-truncated types are called *contractible types*, $(-1)$-truncated types are called *h-propositions*, 0-truncated types are called *h-sets*. Following this, Is-$(-2)$-type is just Contr, Is-$(-1)$-type is just IsHProp and $\mathrm{Type}_{-1}$ is HProp, and Is-0-type is just IsHSet and $\mathrm{Type}_0$ is HSet. An explanation of this terminology might be helpful. Contractible types are types, inhabited by a center $c$, with paths between any point and $c$. A kind of magic thing about contractible types is the lemma

**Lemma 13**
*If* $A : \mathrm{Type}$ *is contractible, then for any* $x, y : A$*, the type* $x = y$ *is contractible.*

Inductively, it means that paths types of any level over a contractible type is contractible. It can be seen as the fact that a contractible type contains only one point $c$, for which there is only one path $c = c$, *etc.* The canonical example of contractible type is $\mathbf{1}$, and actually, any contractible type is equivalent to $\mathbf{1}$. Then, h-propositions are types where any two points are connected by a path. The only difference with contractible types is that we allow the type not to be inhabited. H-propositions are then proof-irrelevant types, in the sense that under the propositions-as-types principle, any points $x$ and $y$ in an HProp are equal, with a unique equality, which is thus irrelevant.

**Lemma 14 : Example of h-propositions**
*The following types are h-propositions:*

- $\mathbf{0}, \mathbf{1}$
- $\mathrm{IsEquiv}(f)$ *for any* $A, B : \mathrm{Type}$ *and* $f : A \to B$.
- $\mathrm{Is}\text{-}n\text{-}\mathrm{type}(A)$ *for any type* $A : \mathrm{Type}$ *and truncation index* $n$.

Now, by definition, h-sets are types for which identity types are in HProp. The world of h-sets can thus be seen as the "usual mathematical" world, as it satisfies proof-irrelevance.

The $n$-truncated types are stable under a lot of operations, as:

**Lemma 15**
*Let* $n \geqslant -2$ *be a truncation index. Then*

- *If* $A : \mathrm{Type}_n$ *and* $x, y : A$, *then* $\mathrm{Is}\text{-}n\text{-}\mathrm{type}(x = y)$.
- *If* $A : \mathrm{Type}_n$ *and* $B : A \to \mathrm{Type}_n$, *then* $\mathrm{Is}\text{-}n\text{-}\mathrm{type}\left(\sum_{x:A} B\, a\right)$.
- *If* $A : \mathrm{Type}$ *and* $B : A \to \mathrm{Type}_n$, *then* $\mathrm{Is}\text{-}n\text{-}\mathrm{type}\left(\prod_{x:A} B\, a\right)$. *Note that the source type needs not to be truncated.*
- *If* $X : \mathrm{Type}_n$, $Y : \mathrm{Type}$ *and* $X \simeq Y$, *then* $\mathrm{Is}\text{-}n\text{-}\mathrm{type}(Y)$.
- *If* $X : \mathrm{Type}_n$, *then* $\mathrm{Is}\text{-}(n+1)\text{-}\mathrm{type}(X)$.
- *We have* $\mathrm{Is}\text{-}(n+1)\text{-}\mathrm{type}\,\mathrm{Type}_n$.

We although note that some types are not $n$-truncated for any $n$ [HoTT, Example 8.8.6] ; it is highly suspected for example that the sphere $\mathbb{S}^2$ is one of these $\infty$-truncated types (it is at least true in homotopy theory).

## Truncations

We present here a way to change any type into a $n$-truncated type, using truncations. The interested reader can read Nicolai Kraus' PhD thesis [Kra15] consecrated to truncation levels in HoTT.

Let $n \geqslant -1$ be a truncation index. The $n$-truncation of a type $A$ is the higher inductive type $\|A\|_n$ generated by

$$\left|\begin{array}{lcl} \mathrm{tr}_n & : & A \to \|A\|_n \\ \alpha^n_{\mathrm{tr}} & : & \text{Is-}n\text{-type}(\|A\|_n) \end{array}\right.$$

If $a : A$, $\mathrm{tr}_n(a)$ will be noted $|a|_n$. The elimination principles are:

**Lemma 16 : Elimination principle of truncations**
*Let $A : \mathrm{Type}$ and $n \geqslant -1$ be a truncation index.*

- *If $P : \mathrm{Type}$ such that $\text{Is-}n\text{-type}(P)$ and $f : A \to P$, then there is a map*

  $$|f|_n : \|A\|_n \to P$$

  *such that for all $a : A$, $|f|_n(|a|_n) \equiv f(a)$.*

- *If $P : \|A\|_n \to \mathrm{Type}$ such that $\prod_{x:\|A\|_n} \text{Is-}n\text{-type}(P\,x)$ and $f : \prod_{a:A} P(|a|_n)$, then there is a dependent map*

  $$|f|_n : \prod_{x:\|A\|_n} P\,x$$

  *such that for all $a : A$, $|f|_n(|a|_n) \equiv f(a)$.*

Basically, this induction principle says that $\|A\|_n$ has contains as much data about $A$ than $A$ itself, but it can only be used to define a $n$-truncated type. It can be expressed as the following universal property:

**Lemma 17 : Universal property of truncations**
*Let $A : \mathrm{Type}$ and $P : \mathrm{Type}_n$. Then the map*

Check post or pre

$$\mathrm{precompose}_{\mathrm{tr}_n} : \begin{array}{ccc} A \to P & \longrightarrow & \|A\|_n \to P \\ f & \longmapsto & f \circ \mathrm{tr}_n \end{array}$$

*is an equivalence.*

We will see in chapter 3 that all $(\text{Is-}n\text{-type}, \|\bullet\|_n)$ define *modalities*.

Truncations : writ e me

$\|A\|_n$: $n$-truncation of $A$

# 3 Higher modalities

As said in the introduction, the main purpose of our work is to build, from a model $\mathfrak{M}$ of homotopy type theory, another model $\mathfrak{M}'$ satisfying new principles. Of course, $\mathfrak{M}'$ should be describable *inside* $\mathfrak{M}$. In set theory, it corresponds to building *inner models* ([Kun]).In type theory, it can be rephrased in terms of left-exact modalities: it consists of an operator $\bigcirc$ on types such that for any type $A$, $\bigcirc A$ satisfies a desired property. If the operator has a "good" behaviour, then it is a modality, and the universe of all types satisfying the chosen property forms a new model of homotopy type theory.

## 3.1 Modalities

**Definition 18**
*A left exact modality is the data of*

*(i) A predicate* $P : \mathrm{Type} \to \mathrm{HProp}$

*(ii) For every type* $A$*, a type* $\bigcirc A$ *such that* $P(\bigcirc A)$

*(iii) For every type* $A$*, a map* $\eta_A : A \to \bigcirc A$

*such that*

*(iv) For every types* $A$ *and* $B$*, if* $P(B)$ *then*

$$\begin{cases} (\bigcirc A \to B) & \to & (A \to B) \\ f & \mapsto & f \circ \eta_A \end{cases}$$

*is an equivalence.*

*(v) for any* $A : \mathrm{Type}$ *and* $B : A \to \mathrm{Type}$ *such that* $P(A)$ *and* $\prod_{x:A} P(Bx)$*, then* $P\left(\sum_{x:A} B(x)\right)$

*(vi) for any* $A : \mathrm{Type}$ *and* $x, y : A$*, if* $\bigcirc A$ *is contractible, then* $\bigcirc(x = y)$ *is contractible.*

*Conditions (i) to (iv) define* a reflective subuniverse*, (i) to (v)* a modality.

**Proposition 19**
*Here are the properties of modalities.*

## 3.2 Examples of modalities

### The identity modality

Let us begin with the most simple modality one can imagine: the one doing nothing. We can define it by letting $\bigcirc A \overset{def}{=} A$ for any type $A$, and $\eta_A \overset{def}{=}$ idmap. Obviously, the desired computation rules are satisfied, so that the identity modality is indeed a left-exact modality.

It might sound useless to consider such a modality, but it can be precious when looking for properties of modalities: if it does not hold for the identity modality, it cannot hold for an abstract one.

### Truncations

The first class of non-trivial examples might be the *truncations* modalities.

### Double negation modality

The double negation modality $\bigcirc A \overset{def}{=} \neg\neg A$ is a modality. Unfortunately, it appears that every type is collapsed to an HProp, thus it cannot be used as-is. The main purpose of this thesis, in particular chapter 5 is to extend this modality into a better one.

Finish modalities

## 3.3 New type theories

**Proposition 20**

## 3.4 Truncated modalities

## 3.5 Translation

# 4 Colimits

> A comathematician is a device turning cotheorems into ffee.
>
> Mathematical folklore

As seen in chapter 2, adding sigma-types to type theory results in adding limits over graphs in the underlying category, and adding higher inductives typves results in adding colimits over graphs. If limits has been extensively studied in [AKL15], theory of colimits was not completely treated.

The following is conjoint work with Simon Boulier and Nicolas Tabareau, helped by precious discussions with Egbert Rijke. The sections 4.1 and 4.2 are extended version of the blog post [Bou16].

## 4.1 Colimits over graphs

As colimits are just dual to limits, it seems that it would be very easy to translate the work on limits to colimits. Althought, even if it might be because we are more habituated to manipulate sigma-types than higher inductive types, it seems way harder.

### Definitions

Let's recall the definitions of graphs and diagrams over graphs, introduced in [AKL15].

**Definition 21 : Graph**
*A* graph *$G$ is the data of*

- *a type $G_0$ of vertices ;*
- *for any $i, j : G_0$, a type $G_1(i, j)$ of edges.*

**Definition 22 : Diagram**
*A* diagram *$D$ over a graph $G$ is the data of*

- *for any $i : G_0$, a type $D_0(i)$ ;*
- *for any $i, j : G_0$ and all $\phi : G_1(i, j)$, a map $D_1(\phi) : D_0(i) \to D_0(j)$*

When the context is clear, $G_0$ will be simply denoted $G$, $G_1(i,j)$ will be noted $G(i,j)$, $D_0(i)$ will be noted $D(i)$ or $D_i$, and $D_1(\phi)$ will be noted $D_{i,j}(\phi)$ or simply $D(\phi)$ ($i$ and $j$ can be inferred from $\phi$).

Examples :

- One can consider the following graph, namely the graph of (co)equalizers

  $$\begin{tikzcd} \bullet \arrow[r, shift left] \arrow[r, shift right] & \bullet \end{tikzcd}$$

  Here, $G_0 = \mathbf{2}$, $G_1(\top,\bot) = \mathbf{2}$ and other $G_1(i,j)$ are empty.

  A diagram over this graph consists of two types $A$ and $B$, and two maps $f, g : A \to B$, producing the diagram

  $$\begin{tikzcd} A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \end{tikzcd}$$

- The graph of the mapping telescope is

  $$\begin{tikzcd} \bullet \arrow[r] & \bullet \arrow[r] & \cdots \end{tikzcd}$$

  In other words, $G_0 = \mathbb{N}$ and $G_1(i, i+1) = \mathbf{1}$.

  A diagram over the mapping telescope is a sequence of types $P : \mathbb{N} \to \text{Type}$ together with arrows $f_n : P_n \to P_{n+1}$:

  $$\begin{tikzcd} P_0 \arrow[r, "f_0"] & P_1 \arrow[r, "f_1"] & \cdots \end{tikzcd}$$

What we would like now would be to define the colimits of these diagrams over graphs, that would satisfy type theoretic versions of usual properties: it should make the diagram commute, and be universal with respect to this property. From now on, let $G$ be a graph and $D$ a diagram over this graph.

The commutation of the diagram is easy: the colimit should be the tip of a cocone.

**Definition 23 : Cocone**
*Let $Q$ be a type. A cocone over $D$ intro $Q$ is the data of arrows $q_i : D_i \to Q$, and for any $i, j : G$ and $g : G(i,j)$, an homotopy $q_j \circ D(g) \sim q_i$.*

If $Q$ and $Q'$ are type with an arrow $f : Q \to Q'$, and if $C$ is a cocone over $D$ into $Q$, one can easily build a cocone on $D$ into $Q'$ by postcomposing all maps of the cocone by $f$, giving a map

$$\text{postcompose}_{\text{cocone}} : \text{cocone}_D(Q) \to (Q' : \text{Type}) \to (Q \to Q') \to \text{cocone}_D(Q')$$

The other way around (from a cocone into $Q'$, give a map $Q \to Q'$) is exactly the second condition we seek:

**Definition 24 : Universality of a cocone**
*Let $Q$ be a type, and $C$ be a cocone over $D$ into $Q$. $C$ is said universal if for any type $Q'$, $\mathrm{postcompose}_{\mathrm{cocone}}(C, Q')$ is an equivalence.*

We can finally define what it means for $Q$ to be a colimit of $D$.

**Definition 25 : Colimit**
*A type $Q$ is said to be a colimit of $D$ if there is a cocone $C$ over $D$ into $Q$, which is universal.*

Example – Let $A$, $B$ be types and $f, g : A \to B$. Let $Q$ be the HIT generated by $\left| \begin{array}{lcl} q & : & B \to Q \\ \alpha & : & q \circ f \sim q \circ g \end{array} \right.$. Then $Q$ is a colimit of the coequalizer diagram associated to $A, B, f, g$. We say that $Q$ is a coequalizer of $f$ and $g$.

Note that for any diagram $D$, one can build a free colimit of $D$, namely the higher inductive type $\mathrm{colim}(D)$ generated by

$$\left| \begin{array}{lcl} \mathrm{colim} & : & \prod_{i:G} D_i \to \mathrm{colim}(D) \\ \alpha_{\mathrm{colim}} & : & \prod_{ij:G} \prod_{g:G(i,j)} \prod_{x:D_i} \mathrm{colim}_j \circ D(g) \sim \mathrm{colim}_i \end{array} \right.$$

Finish colimits

### Properties of colimits

### Towards highly coherent colimits

## 4.2 Van Doorn's and Boulier's constructions

**Proposition 26**

## 4.3 Towards groupoid objects

# 5 Sheaves in homotopy type theory

> Reductio ad absurdum, which Euclid loved so much, is one of a mathematician's finest weapons. It is a far finer gambit than any chess gambit: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game
>
> G.H. Hardy

Section 5.1 briefly recalls the topos-theoretic version of Lawvere-Tierney sheaves theory, and section 5.2 presents the main result of this thesis: the construction of the sheafification modality in homotopy type theory. Section 5.3 tries to link our construction with forcing in type theory.

## 5.1 Sheaves in topoi

In this section, we will rather work in an arbitrary topos rather in type theory. The next section will present a generalisation of the results presented here.

Let us fix for the whole section a topos $\mathscr{E}$, with subobject classifier $\Omega$. A *Lawvere-Tierney topology* on $\mathscr{E}$ is a way to modify slightly truth values of $\mathscr{E}$. It allows to speak about *locally true* things instead of *true* things.

**Definition 27 : Lawvere-Tierney topology [MM92]**
*A Lawvere-Tierney topology is an endomorphism $j : \Omega \to \Omega$ preserving $\top$ ($j\,\top = \top$), idempotent ($j \circ j = j$) and commuting with products ($j \circ \wedge = \wedge \circ (j, j)$).*

A classical example of Lawvere-Tierney topology is given by double negation. Other examples are given by Grothedieck topologies, in the sense

**Theorem 28 : [MM92, p. V.1.2]**
*Every Grothendieck topology $J$ on a small category $\mathbf{C}$ determines a Lawvere-Tierney topology $j$ on the presheaf topos $\mathbf{Sets}^{\mathbf{C}^{\mathrm{op}}}$.*

Any Lawvere-Tierney topology $j$ on $\mathscr{E}$ induces a closure operator $A \mapsto \overline{A}$ on subobjects. If we see a subobject $A$ of $E$ as a characteristic function $\chi_A$, the closure $\overline{A}$ corresponds to the subobject of $E$ whose characteristic function is

$$\chi_{\overline{A}} = j \circ \chi_A.$$

A subobject $A$ of $E$ is said to be dense when $\overline{A} = E$.

Then, we are interested in objects of $\mathscr{E}$ for which it is impossible to make a distinction between objects and their dense subobjects, *i.e* for which "true" and "locally true" coincide. Such objects are called *sheaves*, and are defined as

**Definition 29 : Sheaves[MM92, p. V.2]**
*On object $F$ of $\mathscr{E}$ is a sheaf (or $j$-sheaf) if for every dense monomorphism $m : A \hookrightarrow E$ in $\mathscr{E}$, the canonical map* $\mathrm{Hom}_{\mathscr{E}}(E, F) \to \mathrm{Hom}_{\mathscr{E}}(A, F)$ *is an isomorphism.*

One can show that $\mathrm{Sh}_{\mathscr{E}}$, the full sub-category of $\mathscr{E}$ given by sheaves, is again a topos, with classifying object

$$\Omega_j = \{P \in \Omega \mid jP = P\}.$$

Lawvere-Tienrey sheafification is a way to build a left adjoint $\mathbf{a}_j$ to the inclusion $\mathscr{E} \hookrightarrow \mathrm{Sh}_{(}\mathscr{E})$, exhibiting $\mathrm{Sh}_{(}\mathscr{E})$ as a reflective subcategory of $\mathscr{E}$. In particular, that implies that logical principles valid in $\mathscr{E}$ are still valid in $\mathrm{Sh}_{(}\mathscr{E})$.

For any object $E$ of $\mathscr{E}$, $\mathbf{a}_j(E)$ is defined as in the following diagram

$$\begin{tikzcd}
E \arrow[rr, "\{\cdot\}_E"] \arrow[d, "\theta_E"'] & & \Omega^E \arrow[d, "j^E"] \\
E' \arrow[rr, hook] \arrow[dr, "\text{closure}"'] & & \Omega_j^E \\
& \mathbf{a}_j(E) \arrow[ur] &
\end{tikzcd}$$

The proof that $\mathbf{a}_j$ defines a left adjoint to the inclusion can be found in [MM92].

One classical example of use of sheafification is the construction, from any topos, of a boolean topos negating the continuum hypothesis. More precisely:

**Theorem 30 : Negation of CH [MM92, p. VI.2.1]**
*There exists a Boolean topos satisfying the axiom of choice, in which the continuum hypothesis fails.*

The proof actually follows almost exactly the famous proof of the construction by Paul Cohen of a model of ZFC negating the continuum hypothesis [CD66]. Together with the model of constructible sets $\mathfrak{L}$ by Kurt Gödel [Göd40], it proves that CH is independent of ZFC, solving first Hilbert's problem.

## 5.2 Sheaves in homotopy type theory

The idea of this section is to consider sheafification in topoi as only the first step towards sheafification in type theory. We note that axioms for a Lawvere-Tierney topology on the subobject classifier $\Omega$ of a topos are very close to those of a modality on $\Omega$. We will extensively use this idea, applying it to every subobject classifier $\mathrm{Type}_n$ we described in 2.4. The subobject classifier $\Omega$ of a topos is seen as the *truth values* of the topos, which corresponds to the type HProp in our setting ; the topos is considered proof irrelevant, corresponding to our HSet. Sheafification in topoi is thus a way, when translated to the setting of homotopy type theory, to build, from a left-exact modality on HProp, a left-exact modality on HSet. Our hope in this section is to iterate this construction by applying it to the subobject classifier HSet equipped with a left-exact modality, to build a new left-exact modality on $\mathrm{Type}_1$, and so on.

Fix the reference

The first thing we can note is that such a construction will not allow to reach every type: it is known that there exists types with no finite truncation level [HoTT, Example 8.8.6]. Even worse, some types are not even the limit of its successive truncations, even in a hypercomplete setting [MV98]. It suggests that defining a sheafification functor for all truncated types won't give (at least easily) a sheafification functor on whole Type. Another issue that can be pointed is the complexity of proofs. If, in a topos-theoretic setting, everything is proof-irrelevant, it won't be the case for higher settings, forcing us to prove results that were previously true on the nose. This will oblige us to write long and technical proofs of coherence, and more deeply, to modify completely some lemmas, such as Proposition [MM92, p. IV.7.8], stating that epimorphisms are coequalizers of their kernel pair.

The main idea is thus to follow as closely as possible the topos-theoretic construction, and change as few times as possible to make it work in our higher setting.

Note that the principles we want to add are added directly from the HProp level, the extension to all truncated types is automatic. The choice of the left-exact modality on HProp is thus crucial. For the rest of the section, we fix one, note $\bigcirc_{-1}$. The reader can think of the double negation $\bigcirc_{\neg\neg}$ defined in 3.2. We will define, by induction on the truncation level, left-exact modalities on all $\mathrm{Type}_n$, as in the following theorem.

**Theorem 31**
*The sequence defined by induction by*

$$\begin{aligned} &\bigcirc : \forall\ (n : nat),\ \mathrm{Type}_n \to \mathrm{Type}_n \\ &\bigcirc_{-1}\ (T) \overset{def}{=} j\,T \\ &\bigcirc_{n+1}(T) \overset{def}{=} \sum_{u:T\to\mathrm{Type}_n^{\bigcirc}} \bigcirc_{-1} \left\| \sum_{a:T} u = (\lambda t,\ \bigcirc_n(a=t)) \right\| \end{aligned}$$

*defines a sequence of left-exact modalities, coherent with each others in the* *sense that the following diagram commutes for any $P$ :* Type$_n$*, where $\hat{P}$ is $P$ seen as an inhabitant of* Type$_{n+1}$*.*

$$\begin{tikzcd}
P \arrow[r, "\sim"] \arrow[d, "\eta_n"'] & \widehat{P} \arrow[d, "\eta_{n+1}"] \\
\bigcirc_n P \arrow[r, "\sim"] & \bigcirc_{n+1} \widehat{P}
\end{tikzcd}$$

### Sheaf theory

Let $n$ be a truncation index greater that $-1$, and $\bigcirc_n$ be the left-exact modality given by our induction hypothesis. As in the topos-theoretic setting, we will define what it means for a type to be a $n$-sheaf (or just "sheaf" if the context is clear), and consider the reflective subuniverses of these sheaves ; the reflector will exactly be the sheafification functor. The main issue to give the "good" definition is the choice of the subobject classifier in which dense subobjects will be chosen: two choices appears, HProp and Type$_n$ ; we will actually use both. What guided our choice is the crucial property that the type of all $n$-sheaves has to be a $(n+1)$-sheaf.

From the modality $\bigcirc_n$, one can build a *closure operator*.

**Definition 32**
*Let $E$ be a type.*

- *The* closure *of a subobject of $E$ with $n$-truncated homotopy fibers (or $n$-subobject of $E$, for short), classified by $\chi : E \to$* Type$_n$*, is the subobject of $E$ classified by $\bigcirc_n \circ \chi$.*
- *An $n$-subobject of $E$ classified by $\chi$ is said to be* closed in $E$ *if it is equal to its closure,* i.e *if $\chi = \bigcirc_n \circ \chi$.*
- *An $n$-subobject of $E$ classified by $\chi$ is said to be* dense in $E$ *if its closure is $E$,* i.e *if $\bigcirc_n \circ \chi = \lambda e, \mathbf{1}$*

Topos-theoretic sheaves are characterized by a property of existence and uniqueness, which will be translated, as usual, into a proof that a certain function is an equivalence.

**Definition 33 : Restriction**
*Let $E, F$ :* Type *and $\chi : E \to$* Type*. We define the* restriction map $\Phi_E^\chi$ *as follows*

$$\Phi_E^\chi : \begin{array}{ccc} E \to F & \longrightarrow & \sum_{e:E} \chi e \to F \\ f & \longmapsto & f \circ \pi_1 \end{array} .$$

Here, we need to distinguish between dense $(-1)$-subobjects, that will be used in the definition of sheaves, and dense $n$-subobjects, that will be used in the definition of separated types.

**Definition 34 : Separated Type**
*A type $F$ in* $\mathrm{Type}_{n+1}$ *is* separated *if for any type $E$, and all dense $n$-subobject of $E$ classified by $\chi$, $\Phi_E^\chi$ is an embedding.*

With topos theory point of view, it means that given a map $\sum_{e:E} \chi\, e \to F$, if there is an extension $\tilde{f} : E \to F$, then it is unique, as in

$$\begin{tikzcd}
\sum_{e:E} \chi e \arrow[r, "f"] \arrow[d, "\pi_1"'] & F \\
E \arrow[ur, dashed, "!"'] &
\end{tikzcd}$$

**Definition 35 : Sheaf**
*A type $F$ of* $\mathrm{Type}_{n+1}$ *is a* $(n+1)$-sheaf *if it is separated, and for any type $E$ and all dense $(-1)$-subobject of $E$ classified by $\chi$, $\Phi_E^\chi$ is an equivalence.*

In topos-theoretic words, it means that given a map $f : \sum_{e:E} \chi\, e \to F$, one can extend it uniquely to $\tilde{f} : E \to F$, as in

$$\begin{tikzcd}
\sum_{e:E} \chi e \arrow[r, "f"] \arrow[d, "\pi_1"'] & F \\
E \arrow[ur, dashed, "\exists!"'] &
\end{tikzcd}$$

Note that these definitions are almost the same as the ones in [MM92]. The main difference is that separated is defined for $n$-subobjects, while sheaf only for $(-1)$-subobjects. It might seem bizarre to make such a distinction, but the following proposition gives a better understanding of the situation.

**Proposition 36**
*A type $F$ is* $\mathrm{Type}_{n+1}$ *is separated if, and only if all its path types are $n$-modal, ie*

$$\prod_{x,y:F} (\bigcirc_n(x = y)) = (x = y).$$

A $(n+1)$-sheaf is hence just a type satisying the usual property of sheaves (*i.e* existence of uniqueness of arrow extension from dense $(-1)$-subobjects), with the condition that all its path types are $n$-sheaves. It is a way to force the compatibility of the modalities we are defining.

On can check that the property IsSeparated (resp. IsSheaf) is HProp: given a $X : \mathrm{Type}_{n+1}$, there is only one way for it to be separated (resp. a sheaf). In particular, when needed to prove equality between two sheaves, it suffices to show the equality between the underlying types.

As said earlier, these definitions allow us to prove the fundalemental property that the type of all $n$-sheaves is itself a $(n+1)$-sheaf (this can be viewed as an equivalent definition of left-exactness).

**Proposition 37**
$\mathrm{Type}_n^{\bigcirc}$ *is a* $(n+1)$*-sheaf.*

*Proof.* We have two things to prove here : separation, and sheafness.

- Let $E$ : Type and $\chi : E \to \mathrm{Type}$, dense in $E$. Let $\phi_1, \phi_2 : E \to \mathrm{Type}_n^{\bigcirc}$, such that $\phi_1 \circ \pi_1 = \phi_2 \circ \pi_1$ and let $x : E$. We show $\phi_1(x) = \phi_2(x)$ using univalence.

  As $\chi$ is dense, we have a term $m_x : \bigcirc_n(\chi\, x)$. But as $\phi_2(x)$ is modal, we can obtain a term $h_x : \chi\, x$. As $\phi_1$ and $\phi_2$ are equal on $\sum_{e:E} \chi\, e$, we have an arrow $\phi_1(x) \to \phi_2(x)$. The same method leads to an arrow $\phi_2(x) \to \phi_1(x)$, and one can prove that they are each other inverse.

- Now, we prove that $\mathrm{Type}_n^{\bigcirc}$ is a sheaf. Let $E$ : Type and $\chi : E \to \mathrm{HProp}$, dense in $E$. Let $f : \sum_{e:E} \chi\, e \to \mathrm{Type}_n^{\bigcirc}$. We want to extend $f$ into a map $E \to \mathrm{Type}_n^{\bigcirc}$.

  We define $g$ as $g(e) = \bigcirc_n \left(\mathrm{fib}_\phi(e)\right)$, where

  $$\phi : \sum_{b : \sum_{e:E} \chi\, e} (f\, b) \to E$$

  defined by $\phi(x) = (x_1)_1$. Using the following lemma, one can prove that the map $f \mapsto g$ defines an inverse of $\Phi_E^{\chi}$.

  **Lemma 38**
  *Let* $A, B, C : \mathrm{Type}_n$, $f : A \to B$ *and* $g : B \to C$. *Then if all fibers of* $f$ *and* $g$ *are* $n$*-truncated, then*

  $$\left(\prod_{c:C} \bigcirc_n(\mathrm{fib}_{g\circ f}(c))\right) \simeq \bigcirc_n\left(\sum_{w:\mathrm{fib}_g(c)} \bigcirc_n(\mathrm{fib}_f(w_1))\right).$$

  *Proof.* This is just a modal counterpart of the property characterizing fibers of composition of function. ◇

□

Another fundamental property on sheaves we will need is that the type of (dependent) functions is a sheaf as soon as its codomain is a sheaf.

**Proposition 39**
*If $A : \mathrm{Type}_{n+1}$ and $B : A \to \mathrm{Type}_{n+1}$ such that for any $a : A$, $(B\ a)$ is a sheaf, then $\prod_{a:A} B\,a$ is a sheaf.*

*Proof.* Again, when proving equivalences, we will only define the maps. The proofs of section and retraction are technical, not really interesting, and present in the formalisation.

- *Separation:* Let $E : \mathrm{Type}$ and $\chi : E \to \mathrm{Type}_n$ dense in $E$. Let $\phi_1, \phi_2 : E \to \prod_{a:A} B\,a$ equal on $\sum_{e:E} \chi\,e$ *i.e* such that $\phi_1 \circ \pi_1 = \phi_2 \circ \pi_1$. Then for any $a : A$, $(\lambda x : E,\ \phi_1(x,a))$ and $(\lambda x : E,\ \phi_2(x,a))$ coincide on $\sum_{e:E}(\chi\,e)$, and as $B\,a$ is separated, they coincide also on all $E$.

- *Sheaf:* Let $E : \mathrm{Type}$, $\chi : E \to \mathrm{HProp}$ dense in $E$ and $f : \sum_{e:E} \chi\,e \to \prod_{a:A} B\,a$. Let $a : A$ ; the map $(\lambda x,\ f(x,a))$ is valued in the sheaf $B\,a$, so it can be extended to all $E$, allowing $f$ to be extended to all $E$.

□

## Sheafification

The sheafification process will be defined in two steps. The first one will build, from any $T : \mathrm{Type}_{n+1}$, a separated object $\square_{n+1}\,T : \mathrm{Type}_{n+1}$ ; one can show that $\square_{n+1}$ defines a modality on $\mathrm{Type}_{n+1}$. The second step will build, from any separated type $T : \mathrm{Type}_{n+1}$, a sheaf $\bigcirc_{n+1}(T)$ ; one can show that $\bigcirc_{n+1}$ is indeed the left-exact modality we are searching.

Let $n$ be a fixed truncation index, and $\bigcirc_n$ a left-exact modality on $\mathrm{Type}_n$, compatible with $\bigcirc_{-1}$ as in

**Condition 40**
*For any mere proposition $P$ (where $\widehat{P}$ is $P$ seen as a $\mathrm{Type}_n$), $\bigcirc_n\widehat{P} = \bigcirc_{-1}P$ and the following coherence diagram commutes*

$$\begin{tikzcd}
P \arrow[r, "\sim"] \arrow[d, "\eta_{-1}"'] & \widehat{P} \arrow[d, "\eta_n"] \\
\bigcirc_{-1}P \arrow[r, "\sim"] & \bigcirc_n\widehat{P}
\end{tikzcd}$$

### From types to separated types

Let $T : \mathrm{Type}_{n+1}$. We define $\square_{n+1} T$ as the image of $\bigcirc_n^T \circ \{\cdot\}_T$, as in

$$\begin{tikzcd}
T \arrow[r, "\{\cdot\}_T"] \arrow[d, "\mu_T"'] & (\mathrm{Type}_n)^T \arrow[d, "\bigcirc_n^T"] , \\
\square_{n+1} T \arrow[r] & \left(\mathrm{Type}_n^{\bigcirc}\right)^T
\end{tikzcd}$$

where $\{\cdot\}_T$ is the singleton map $\lambda(t : T),\ \lambda(t' : T),\ t = t'$. $\square_{n+1} T$ can be given explicitly by

$$\begin{aligned}
\square_{n+1} T &\overset{def}{=} \mathrm{Im}(\lambda\, t : T,\ \lambda\, t',\ \bigcirc_n(t = t')) \\
&\overset{def}{=} \sum_{u : T \to \mathrm{Type}_n^{\bigcirc}} \left\| \sum_{a:A} (\lambda t,\ \bigcirc_n(a = t)) = u \right\|.
\end{aligned}$$

This corresponds to the free separated object used in the topos-theoretic construction, but using $\mathrm{Type}_n^{\bigcirc}$ instead of the $j$-subobject classifier $\Omega_j$.

**Proposition 41**
*For any $T : \mathrm{Type}_{n+1}$, $\square_{n+1} T$ is separated.*

*Proof.* We use the following lemma:

**Lemma 42**
*A $(n+1)$-truncated type $T$ with an embedding $f : T \to U$ into a separated $(n+1)$-truncated type $U$ is itself separated.*

Post or Pre?

*Proof.* Let $E : \mathrm{Type}$ and $\chi : E \to \mathrm{Type}_n$ dense in $E$. Let $\phi_1, \phi_2 : \sum_{e:E} \chi\, e \to T$ such that $\phi_1 \circ \pi_1 \sim \phi_2 \circ \pi_1$. Postcomposing by $f$ yields an homotopy $f \circ \phi_1 \circ \pi_1 \sim f \circ \phi_2 \circ \pi_1$. As $f \circ \phi_1, f \circ \phi_2 : \sum_{e:E} \chi\, e \to U$, and $U$ is separated, we can deduce $f \circ \phi_1 \sim f \circ \phi_2$. As $f$ is an embedding, $\phi_1 \sim \phi_2$. ◇

As $\square_{n+1} T$ embeds in $\left(\mathrm{Type}_n^{\bigcirc}\right)^T$, we only have to show that the latter is separated. But it is the case because $\mathrm{Type}_n^{\bigcirc}$ is a sheaf (by Proposition 37) and a function type is a sheaf as soon as its codomain is a sheaf (by Proposition 39). □

We will now show that $\square_{n+1}$ defines a modality, with unit map $\mu$. The left-exactness of $\bigcirc_{n+1}$ will come from the second part of the process. The first thing to show that $\square_{n+1} T$ is universal among separated type below $T$. In the topos-theoretic sheafification, it comes easily from the fact that epimorphims are coequalizers of their kernel pairs. As it is not true anymore in our setting, we will use its generalization, proposition 26.Here is a sketch of the proof: as $\mu_T$ is a surjection (it is defined by the surjection-embedding factorization), $\square_{n+1} T$ is the colimit of its iterated kernel pair. Hence, for any type $Q$ defining a cocone on $\mathrm{KP}(\mu_T)$, there is a unique arrow $\square_{n+1} T \to Q$. What remains to show is any separated type $Q$ defines a cocone on $\mathrm{KP}(\mu_T)$ ; we will actually show that any separated type $Q$ defines a cocone on $\|\mathrm{KP}(\mu_T)\|_{n+1}$, which is enough. We do it by defining another diagram $\mathring{T}$, equivalent to $\|\mathrm{KP}(\mu_T)\|_{n+1}$, for which it is easy to define a cocone into any separated type $Q$.

This comes from the following construction which connects $\square_{n+1} T$ to the colimit of the iterated kernel pair of $\mu_T$.

**Definition 43**
*Let $X : \mathrm{Type}$. Let $\mathring{T}_X$ be the higher inductive type generated by*

- $\mathring{t} :\ \|X\|_{n+1} \to \mathring{T}_X$
- $\mathring{\alpha} :\ \forall a\, b : \|X\|_{n+1},\ \bigcirc(a = b) \to \mathring{t}(a) = \mathring{t}(b)$
- $\mathring{\alpha}_1 :\ \forall a : \|X\|_{n+1},\ \mathring{\alpha}(a, a, \eta_{a=a} 1) = 1$

*We view $\mathring{T}$ as the coequalizer of*

$$\begin{tikzcd} \sum\limits_{a,b:\|X\|_{n+1}} \bigcirc(a = b) \arrow[r, shift left, "\pi_1"] \arrow[r, shift right, "\pi_2"'] & \|X\|_{n+1} \end{tikzcd}$$

*preserving $\eta_{a=a} 1$.*

*We consider the diagram $\mathring{T}$ :*

$$\begin{tikzcd} \|X\|_{n+1} \arrow[r] & \|\mathring{T}_X\|_{n+1} \arrow[r] & \|\mathring{T}_{\mathring{T}_X}\|_{n+1} \arrow[r] & \cdots \end{tikzcd}$$

The main result we want about $\mathring{T}$ is the following:

**Lemma 44**
*Let $T : \mathrm{Type}_{n+1}$. Then $\square_{n+1} T$ is the $(n+1)$-colimit of the diagram $\mathring{T}$.*

The key point of the proof is that diagrams $\mathring{T}$ and $\|\mathrm{KP}(\mu_T)\|_{n+1}$ are equivalent. We will need the following lemma:

**Lemma 45**
*Let $A, S : \mathrm{Type}_{n+1}$, $S$ separated, and $f : A \to S$. Then if*

$$\forall a, b : A,\ f(a) = f(b) \simeq \bigcirc(a = b), \tag{5.1}$$

*then*

$$\forall a, b : \|\mathrm{KP}_f\|_{n+1},\ |\tilde{f}|_{n+1}(a) = |\tilde{f}|_{n+1}(b) \simeq \bigcirc(a = b).$$

*Sketch of proof.* By induction on truncation, we need to show that

$$\forall a, b : \mathrm{KP}_f,\ \tilde{f}(|a|_{n+1}) = \tilde{f}(|b|_{n+1}) \simeq \bigcirc(|a|_{n+1} = |b|_{n+1}).$$

We use the encode-decode [HoTT, Section 8.9] method to characterize $\tilde{f}(|a|_{n+1}) = x$, and the result follows. We refer to the formalization for details. □

This lemmas allows to prove that, in the iterated kernel pair diagram of $f$

$$\begin{tikzcd}
X \arrow[r] \arrow[drr, "f"'] & \mathrm{KP}(f) \arrow[r] \arrow[dr, "f_1"] & \mathrm{KP}(f_1) \arrow[r] \arrow[d, "f_2"'] & \mathrm{KP}(f_2) \arrow[r] \arrow[dl, "f_3"] & \cdots \\
 & & S & &
\end{tikzcd}$$

if $f$ satisties 5.1, then each $|f_i|_{n+1}$ does.
Nota – It is clear that if $A$ and $B$ are equivalent types, and $\forall a, b : A,\ f(a) = f(b) \simeq \bigcirc(a = b)$, then

$$\mathrm{Coeq}_1\left(\sum_{a,b:A} f\,a = f\,b \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} A\right) \simeq \mathrm{Coeq}_1\left(\sum_{a,b:B} \bigcirc(a = b) \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} B\right)$$

*Proof of lemma 44.* As said, it suffices to show that $\|C(\mu_T)\|_{n+1} = \mathring{T}$.

$$\begin{tikzcd}
\|\mathrm{KP}^0(\mu_T)\|_{n+1} \arrow[r] \arrow[d, "\wr"] & \|\mathrm{KP}^1(\mu_T)\|_{n+1} \arrow[r] \arrow[d, "\wr"] & \|\mathrm{KP}^2(\mu_T)\|_{n+1} \arrow[r] \arrow[d, "\wr"] & \cdots \\
\mathring{T}_0 \arrow[r] & \mathring{T}_1 \arrow[r] & \mathring{T}_2 \arrow[r] & \cdots
\end{tikzcd}$$

The first equivalence is trivial. Let's then start with the second. What we need to show is

$$\|\mathrm{KP}(\mu_T)\|_{n+1} \simeq \|\mathring{T}_T\|_{n+1},$$

*i.e*

$$\mathrm{Coeq}_1\left(\sum_{a,b:T} \mu_T a = \mu_T b \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} T\right) \simeq \mathrm{Coeq}_1\left(\sum_{a,b:T} \bigcirc(a = b) \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} T\right).$$

By the previous remark, it suffices to show that $\mu_T$ satisfies condition (5.1), *i.e* $\forall a, b : T,\ \bigcirc_n(a = b) = (\mu_T a = \mu_T b)$. By univalence, we want arrows in both ways, forming an equivalence.

- Suppose $p : (\mu_T a = \mu_T b)$. Then projecting $p$ along first components yields $q : \prod_{t:T} \bigcirc_n(a = t) = \bigcirc_n(b = t)$. Taking for example $t = b$, we deduce $\bigcirc_n(a = b) = \bigcirc_n(b = b)$, and the latter is inhabited by $\eta_{b=b}1$.

- Suppose now $p : \bigcirc_n(a = b)$. Let $\iota$ be the first projection from $\Box_{n+1} T \to (T \to \mathrm{Type}_n^{\bigcirc})$. $\iota$ is an embedding, thus it suffices to prove $\iota(\mu_T a) = \iota(\mu_T b)$, *i.e* $\prod_{t:T} \bigcirc_n(a = t) = \bigcirc_n(b = t)$. The latter remains true by univalence.

The fact that these two form an equivalence is technical, we refer to the formalization for an explicit proof.

Let's show the other equivalences by induction. Suppose that, for a given $i : \mathbb{N}$, $\|\mathrm{KP}^i(\mu_T)\|_{n+1} \simeq \mathring{T}_i$. We want to prove $\|\mathrm{KP}^{i+1}(\mu_T)\|_{n+1} \simeq \mathring{T}_{i+1}$, *i.e*

$$\left\| \mathrm{Coeq}_1\left( \sum_{a,b:\mathrm{KP}^i(\mu_T)} f_i a = f_i b \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} \mathrm{KP}^i(\mu_T) \right) \right\|_{n+1}$$

$$\simeq \left\| \mathrm{Coeq}_1\left( \sum_{a,b:\|\mathring{T}_i\|_{n+1}} \bigcirc(a = b) \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} \|\mathring{T}_i\|_{n+1} \right) \right\|_{n+1}$$

where $f_i$ is the map $\mathrm{KP}^i(\mu_T) \to \Box_{n+1} T$. But lemma 45 just asserted that $f_i$ satisfies 5.1, hence the previous nota yields the result.

One would need to show that, modulo these equivalences, the arrows of the two diagrams are equal. We leave that to the reader, who can refers to the formalization if needed. □

Now, let $Q$ be any separated $\mathrm{Type}_{n+1}$, and $f : X \to Q$. Then the following diagram commutes

$$\begin{tikzcd}
\|X\|_{n+1} \arrow[r] \arrow[dr] & \|\mathring{T}_X\|_{n+1} \arrow[r] \arrow[d] & \|\mathring{T}_{\mathring{T}_X}\|_{n+1} \arrow[r] \arrow[dl] & \cdots \\
& Q & &
\end{tikzcd}$$

But we know (lemma 44) that $\Box_{n+1} T$ is the $(n+1)$-colimit of the diagram $\mathring{T}$, thus there is an universal arrow $\Box_{n+1} T \to Q$. This is enough to state the following proposition.

**Proposition 46**
$(\Box_{n+1}, \mu)$ *defines a reflective subuniverse on* $\mathrm{Type}_{n+1}$.

To show that $\Box_{n+1}$ is a modality, it remains to show that separation is a property stable under sigma-types. Let $A : \mathrm{Type}_{n+1}$ be a separated type and $B : A \to \mathrm{Type}_{n+1}$ be a family of separated types. We want to show that $\sum_{x:A} Bx$ is separated. Let $E$ be a type, and $\chi : E \to \mathrm{Type}_n$ a dense subobject of E.

Let $f,g$ be two maps from $\sum_{e:E} \chi\, e$ to $\sum_{x:A} B x$, equal when precomposed with $\pi_1$.

$$\begin{tikzcd}
\sum_{e:E} \chi\, e \arrow[r, shift left, "f \circ \pi_1"] \arrow[r, shift right, "g \circ \pi_1"'] \arrow[d, "\text{dense}"'] & \sum_{x:A} B x \\
E \arrow[ur, shift left, "f"] \arrow[ur, shift right, "g"'] &
\end{tikzcd}$$

We can restrict the previous diagram to

$$\begin{tikzcd}
\sum_{e:E} \chi\, e \arrow[r, shift left, "\pi_1 \circ f \circ \pi_1"] \arrow[r, shift right, "\pi_1 \circ g \circ \pi_1"'] \arrow[d, "\text{dense}"'] & A \\
E \arrow[ur, shift left, "\pi_1 \circ f"] \arrow[ur, shift right, "\pi_1 \circ g"'] &
\end{tikzcd}$$

and as $A$ is separated, $\pi_1 \circ f = \pi_1 \circ g$. For the second components, let $x : E$. Notice that $\sum_{y:E} x = y$ has a dense $n$-subobject, $\sum_{y:\sum_{e:E} \chi\, e} x = y_1$:

$$\begin{tikzcd}
\sum_{y:\sum_{e:E} \chi\, e} x = y_1 \arrow[r, shift left, "\pi_2 \circ f \circ \pi_1 \circ \pi_1"] \arrow[r, shift right, "\pi_2 \circ g \circ \pi_1 \circ \pi_1"'] \arrow[d, "\text{dense}"'] & B x \\
\sum_{y:E} x = y \arrow[ur, shift left, "\pi_2 \circ f \circ \pi_1"] \arrow[ur, shift right, "\pi_2 \circ g \circ \pi_1"'] &
\end{tikzcd}$$

Using the separation property of $B x$, one can show that second components, transported correctly along the first components equality, are equal. The complete proof can be found in the formalization. This proves the following proposition

**Proposition 47**
$(\square_{n+1}, \mu)$ *defines a modality on* $\mathrm{Type}_{n+1}$.

As this modality is just a step in the construction, we do not need to show that it is left exact (actually, it is not), we will only do it for the sheafification modality.

### From Separated Type to Sheaf

For any $T$ in $\mathrm{Type}_{n+1}$, $\bigcirc_{n+1} T$ is defined as the closure of $\square_{n+1} T$, seen as a subobject of $T \to \mathrm{Type}_n^{\bigcirc}$. $\bigcirc_{n+1} T$ can be given explicitly by

$$\bigcirc_{n+1} T \overset{def}{=} \sum_{u:T \to \mathrm{Type}_n^{\bigcirc}} \bigcirc_{-1} \left\| \sum_{a:T} (\lambda t,\ \bigcirc_n (a = t)) = u \right\|.$$

To prove that $\bigcirc_{n+1} T$ is a sheaf for any $T : \mathrm{Type}_{n+1}$, we use the following lemma.

**Lemma 48**
*Any closed $(-1)$-subobject of a sheaf is a sheaf.*

*Proof.* Let $U$ be a sheaf, and $\kappa : U \to \mathrm{HProp}$ be a closed $(-1)$-subobject. Let $E$ : Type and $\chi : E \to \mathrm{HProp}$ dense in $E$. Let $\phi : \sum_{e:E} \chi\, e \to \sum_{u:U} \kappa\, u$. As $\pi_1 \circ \phi$ is a map $\sum_{e:E} \chi\, e \to U$ and $U$ is a sheaf, it can be extended into $\psi : E \to U$. As $\kappa$ is closed, it suffices now to prove $\prod_{e:E} \bigcirc_n(\kappa\,(\psi\, e))$ to obtain a map $E \to \sum_{u:U} \kappa\, u$.

Let $e : E$. As $\chi$ is dense, we have a term $w : \bigcirc_n(\chi\, e)$, and by $\bigcirc_n$-induction, a term $\widetilde{w} : \chi\, e$. Then, by retraction property, $\psi(e) = \phi(e, \widetilde{w})$, and by $\pi_2 \circ \phi$, we have hence our term of type $\kappa(\psi\, e)$.

□

As $T \to \mathrm{Type}_n^{\bigcirc}$ is a sheaf, and $\bigcirc_{n+1} T$ is closed in $T \to \mathrm{Type}_n^{\bigcirc}$, $\bigcirc_{n+1} T$ is a sheaf. We now prove that it forms a reflective subuniverse.

**Proposition 49**
*$(\bigcirc_{n+1}, \nu)$ defines a reflective subuniverse.*

*Proof.* Let $T, Q : \mathrm{Type}_{n+1}$ such that $Q$ is a sheaf. Let $f : T \to Q$. Because $Q$ is a sheaf, it is in particular separated; thus we can extend $f$ to $\square_{n+1} f : \square_{n+1} T \to Q$.

But as $\bigcirc_{n+1} T$ is the closure of $\square_{n+1} T$, $\square_{n+1} T$ is dense into $\bigcirc_{n+1} T$, so the sheaf property of $Q$ allows to extend $\square_{n+1} f$ to $\bigcirc_{n+1} f : \bigcirc_{n+1} T \to Q$.

As all these steps are universal, the composition is. □

Then:

**Proposition 50**
*$(\bigcirc_{n+1}, \nu)$ defines a modality.*

*Proof.* The proof use the same ideas as in subsection 5.2. Let $A : \mathrm{Type}_{n+1}$ a sheaf and $B : A \to \mathrm{Type}_{n+1}$ a sheaf family. By proposition 47, we already know that $\sum_{a:A} B\, a$ is separated. Let $E$ be a type, and $\chi : E \to \mathrm{HProp}$ a dense subobject. Let $f : \sum_{e:E} \chi\, e \to \sum_{x:A} B\, x$ ; we want to extend it into a map $E \to \sum_{x:A} B\, x$.

$$\begin{tikzcd}
\sum_{e:E} \chi\, e \arrow[r, "f"] \arrow[d] & \sum_{x:A} B\, x \\
E \arrow[ur, dotted] &
\end{tikzcd}$$

As $A$ is a sheaf, and $\pi_1 \circ f : \sum_{e:E} \chi\, e \to A$, wa can recover an map $g_1 : E \to A$. We then want to show $\prod_{e:E} B(g_1\, e)$. Let $e : E$. As $\chi$ is dense, we have a term $w : \bigcirc_n(\chi\, e)$, and as $B(g_1\, e)$ is a sheaf, we can recover a term $\widetilde{w} : \chi\, e$. Then $g_1(e) = f(e, \widetilde{w})$, and $\pi_2 \circ f$ gives the result. □

It remains to show that $\bigcirc_{n+1}$ is left exact and is compatible with $\bigcirc_{-1}$. To do that, we need to extend the notion of compatibility and show that actually every modality $\bigcirc_{n+1}$ is compatible with $\bigcirc_n$ on lower homotopy types.

**Proposition 51**
*If $T : \mathrm{Type}_n$, then $\bigcirc_{n+1}\widehat{T} = \bigcirc_n T$, where $\widehat{T}$ is $T$ seen as a $\mathrm{Type}_{n+1}$.*

*Proof.* We prove it by induction on $n$:

- For $n = -1$: Let $T : \mathrm{HProp}$. Then

$$\bigcirc_0 \widehat{T} \overset{def}{=} \sum_{u:T\to\mathrm{Type}_n^{\bigcirc}} \bigcirc_{-1} \left\| \sum_{a:T} (\lambda t,\ \bigcirc_{-1}(a=t)) = u \right\|_{-1}$$
$$= \sum_{u:T\to\mathrm{Type}_n^{\bigcirc}} \bigcirc_{-1} \left( \sum_{a:T} (\lambda t,\ \bigcirc_{-1}(a=t)) = u \right)$$

  because the type inside the truncation is already in HProp. Now, let define $\phi : \bigcirc_{-1} T \to \bigcirc_0 T$ by

$$\phi t = (\lambda t', \mathbf{1}; \kappa)$$

  where $\kappa$ is defined by $\bigcirc_{-1}$-induction on $t$. Indeed, as $T$ is an HProp, $(a = t) \simeq \mathbf{1}$. Let $\psi : \bigcirc_0 T \to \bigcirc_{-1} T$ by obtaining the witness $a : T$ (which is possible because we are trying to inhabit a modal proposition), and letting $\psi(u;x) = \eta_T a$. These two maps form an equivalence (the section and retraction are trivial because the equivalence is between mere propositions).

- Suppose now that $\bigcirc_{n+1}$ is compatible with all $\bigcirc_k$ on lower homotopy types. Let $\bigcirc_{n+2}$ be as above, and let $T : \mathrm{Type}_{n+1}$. Then, as $\bigcirc_{n+1}$ is compatible with $\bigcirc_n$, and $(a = t)$ is in $\mathrm{Type}_n$,

$$\bigcirc_{n+2} \widehat{T} = \sum_{u:T\to\mathrm{Type}_{n+1}^{\bigcirc}} \bigcirc_{-1} \left\| \sum_{a:T} (\lambda t,\ \bigcirc_n(a=t)) = u \right\|_{-1}.$$

  It remains to prove that for every $(u, x)$ inhabiting the $\Sigma$-type above, $u$ is in $T \to \mathrm{Type}_n^{\bigcirc}$, *i.e* that for every $t : T$, Is-$n$-type$(u\,t)$. But for any truncation index $p$, the type Is-$p$-type $X$ : HProp is a sheaf as soon as $X$ is, so we can get rid of $\bigcirc_{-1}$ and of the truncation, which tells us that for every $t : T$, $u\,t = \bigcirc_n(a = t) : \mathrm{Type}_n$. □

This proves in particular that $\bigcirc_{n+1}$ is compatible with $\bigcirc_{-1}$ in the sense of condition 40.

The last step is the left-exactness of $\bigcirc_{n+1}$. Let $T$ be in $\mathrm{Type}_{n+1}$ such that $\bigcirc_{n+1} T$ is contractible. Thanks to the just shown compatibility between $\bigcirc_{n+1}$ and $\bigcirc_n$ for $\mathrm{Type}_n$, left exactness means that for any $x, y : T$, $\bigcirc_n(x = y)$ is contractible.

Using a proof by univalence as we have done for proving $\bigcirc_n(a = b) \simeq (\mu_T(a) = \mu_T(b))$ in Proposition 44, we can show that:

**Proposition 52**
*For all $a, b : T$, $\bigcirc_n(a = b) \simeq (\nu_T a = \nu_T b)$.*

As $\bigcirc_{n+1} T$ is contractible, path spaces of $\bigcirc_{n+1} T$ are contractible, in particular $(\nu_T a = \nu_T b)$, which proves left exactness.

Check from here

### Summary

Starting from any left-exact modality $\bigcirc_{-1}$ on HProp, we have defined for any truncation level $n$, a new left-exact modality $\bigcirc_n$ on $\mathrm{Type}_n$, which corresponds to $\bigcirc_{-1}$ when restricted to HProp.

When $\bigcirc_{-1}$ is consistent (in the sense of proposition 20), $\bigcirc_n \mathbf{0} = \bigcirc_{-1} \mathbf{0}$ is also not inhabited, hence the homotopy type theory induced by $\bigcirc_n$ is consistent. In particular, the modality induced by the double negation modality on HProp is consistent.

In topos theory, the topos of Lawvere-Tierney sheaves for the double negation topology is a boolean topos. In homotopy type theory, this result can be expressed as:

**Proposition 53**
*$(\bigcirc_{\neg\neg})_n$, the modality obtained by sheafification of the double negation modality, induces a type theory where the propositional excluded middle law holds.*

Combined with forcing in type theory [JTS12], it should be possible to lift the proof of independence of the continuum hypothesis to a classical setting, which is where the continuum hypothesis is really meaningful. However, we haven't worked out the details and left this for future work.

### Extension to Type

In the previous section, we have defined a (countably) infinite family of modalities $\mathrm{Type}_i \to \mathrm{Type}_i$. One can extend them to whole Type by composing with truncation:

**Lemma 54**
*Let $\bigcirc_i : \mathrm{Type}_i \to \mathrm{Type}_i$ be a modality. Then $\bigcirc \overset{def}{=} \bigcirc_i \circ \|\cdot\|_i : \mathrm{Type} \to \mathrm{Type}$ is a modality in the sense of [HoTT, Section 7.7]*

If $\bigcirc_{-1}$ is the double negation modality on HProp and $i = -1$, $\bigcirc$ is exactly the double negation modality on Type described in 3.2. Chosing $i \geqslant 0$ is a refinement of this double negation modality on Type: it will collapse every type to a $\mathrm{Type}_i$, instead of an HProp.

Obviously, as truncation modalities are not left-exact [HoTT, Exercise 7.11], $\bigcirc$ isn't either. But in the following sense, when restricted to $i$-truncated types, it is:

**Lemma 55**
*Let $A : \mathrm{Type}_i$. Then if $\bigcirc(A)$ is contractible, for any $x, y : A$, $\bigcirc(x = y)$ is contractible.*

*Proof.* For $i$-truncated types, $\bigcirc = \bigcirc_i$, and $\bigcirc_i$ is left-exact. □

The compatibility between the modalities $\bigcirc_n$ and between the modalities $\|\cdot\|_n$ allow us to chose the truncation index as high as desired. Taking it as a non-fixed parameter allows to work in an universe where the new principle (*e.g.* mere excluded middle) is true for any explicit truncated type. Indeed, $i$ can be chosen dynamically along a proof, and thus be increased as much as needed, without changing results for lower truncated types.

By proposition 20, these left-exact modalities induces a consistent type theory. Furthermore, the univalence remains true in this new type theory in the following sense:

**Proposition 56**
*Let $n$ be a given truncation index, and $\bigcirc$ the modality associated to $n$ as defined in lemma 54. Then, for any type $A, B : \mathrm{Type}_n^{\bigcirc}$, if $\varphi$ is the canonical arrow*

$$A = B \to A \simeq B,$$

*then $\mathrm{IsEquiv}(\varphi)$ is modal.*

*Proof.* The first thing to notice is that, if $X$ and $Y$ are modal, and $f : X \to Y$, then the mere proposition $\mathrm{IsEquiv}\, f$ is also modal. Therefore, it suffices to show that both $A = B$ and $A \simeq B$ are modal. By proposition 19, $A = B$ is modal. Moreover, $(A \simeq B) \simeq \sum_{f:A\to B} \mathrm{IsEquiv}\, f$. Therefore, as $A$ and $B$ are modal, $A \simeq B$ is too.

Hence, $\mathrm{IsEquiv}\, \varphi$ is modal. □

We can view sheafification in terms of model of type theory but because of the resulting modality on Type is not left exact, we need to restrict ourselves to a type theory with only one universe. Let $\mathfrak{M}$ be a model of homotopy type theory with one universe. Using the modality $\bigcirc_{\neg\neg}$ (for any level $n$) associated to the sheafification, there is a model $\bigcirc_{\neg\neg}\mathfrak{M}$ of type theory with one universe (using results in Section 3.3), where excluded middle is true, and which is univalent (as shown in Proposition 56).

## 5.3 Forcing in type theory

Try to write something smart about connections with forcing

# 6 Conclusion and future works

Write a conclusion

# What remains to be done

Write the abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . i
Write the acknowledgments . . . . . . . . . . . . . . . . . . . . . . . . . . iii
Write the intro . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
Write an intro to chapter HoTT . . . . . . . . . . . . . . . . . . . . . . . 3
These references were added without even opening them. Maybe a check would be good . . . . . . . . . . . . . . . . . . . . . . . . . . 15
Check post or pre . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
Truncations : writ e me . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
Finish modalities . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
Finish colimits . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
Fix the reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Post or Pre? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
Check from here . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
Try to write something smart about connections with forcing . . . . . . 41
Write a conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

# List of Notations

| | |
|---|---|
| $(x : A) \to (B\,a)$ | Dependent product over $B$, page 5 |
| $p \cdot q$ | Concatenation of paths, page 7 |
| idpath or 1 | Constant path, page 6 |
| idpath$_x$ or $1_x$ | Constant path over$x$, page 6 |
| $p^{-1}$ | Inverse of path, page 7 |
| $\mathbb{N}$ | Type of naturals, page 5 |
| $\mathbf{1}$ | Unit type, page 4 |
| $\prod_{a:A} B\,a$ | Dependent product over $B$, page 5 |
| $\Sigma A$ | Suspension of a type, page 13 |
| $\simeq$ | Equivalence of types, page 9 |
| $\sum_{a:A} B\,a$ | Dependent sum over $B$, page 5 |
| $\lVert \cdot \rVert$ | $n$-truncation of types, page 17 |
| $\lvert \cdot \rvert_n$ | $n$-truncation of terms or arrows, page 17 |
| $\mathbf{0}$ | Empty type, page 4 |
| $A + B$ | Coproduct of types, page 4 |
| $a : A$ | Judgement “$a$ is of type $A$”, page 3 |
| $a =_A b$ | Type of paths from $a$ to $b$ in $A$, page 6 |
| $a = b$ | Type of paths from $a$ to $b$, page 6 |
| $A \times B$ | Product of types, page 5 |
| $A \to B$ | Type of arrows from $A$ to $B$, page 5 |

# Bibliography

[AKL15] Jeremy Avigad, Krzysztof Kapulkin, and Peter LeFanu Lumsdaine. “Homotopy limits in type theory”. In: *Mathematical Structures in Computer Science* (Published online: 19 january 2015).

[BL11] Andrej Bauer and Peter LeFanu Lumsdaine. *A Coq proof that Univalence Axioms implies Functional Extensionality*. 2011. URL: https://github.com/andrejbauer/Homotopy/raw/master/OberwolfachTutorial/univalence.pdf.

[Bou16] Simon Boulier. *Colimits in HoTT*. Blog post. 2016. URL: http://homotopytypetheory.org/2016/01/08/colimits-in-hott/.

[CD66] P.J. Cohen and M. Davis. *Set theory and the continuum hypothesis*. WA Benjamin New York, 1966.

[Göd40] Kurt Gödel. *The Consistency of the Axiom of Choice and of the Generalized Continuum Hypothesis with the Axioms of Set Theory*. Princeton University Press, 1940.

[HoTT] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. IAS: http://homotopytypetheory.org/book, 2013.

[HoTT/Coq] The Univalent Foundations Program. *Coq HoTT library*. URL: https://github.com/HoTT/HoTT/.

[How80] William A. Howard. “The formulas-as-types notion of construction”. In: *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Ed. by J. P. Seldin and J. R. Hindley. Reprint of 1969 article. Academic Press, 1980, pp. 479–490.

[HS96] Martin Hofmann and Thomas Streicher. “The Groupoid Interpretation of Type Theory”. In: *In Venice Festschrift*. Oxford University Press, 1996, pp. 83–111.

[HTS] Vladimir Voevodsky. *A simple type system with two identity types*. Started in 2013, in progress.

[Hut11] Michael Hutchings. *Introduction to higher homotopy groups and obstruction theory*. 2011. URL: https://math.berkeley.edu/~hutching/.

[JTS12] Guilhem Jaber, Nicolas Tabareau, and Matthieu Sozeau. "Extending type theory with forcing". In: *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*. IEEE. 2012, pp. 395–404.

[Kra15] Nicolai Kraus. "Truncation Levels in Homotopy Type Theory". PhD thesis. University of Nottingham, June 2015.

[Kun] Kenneth Kunen. *Set theory*. Ed. by North-Holland.

[Lic] Dan Licata. *Another proof that univalence implies function extensionality*. Blog post. URL: `http://homotopytypetheory.org/2014/02/17/another-proof-that-univalence-implies-function-extensionality/`.

[LS13] Daniel R. Licata and Michael Shulman. "Calculating the Fundamental Group of the Circle in Homotopy Type Theory". In: *LICS 2013: Proceedings of the Twenty-Eighth Annual ACM/IEEE Symposium on Logic in Computer Science*. 2013.

[m31] Andrej Bauer. *Andromeda implementation*.

[Mar98] Per Martin-Löf. "An intuitionistic theory of types". In: *Twenty-five years of constructive type theory* 36 (1998), pp. 127–172.

[MM92] Saunders MacLane and Ieke Moerdijk. *Sheaves in Geometry and Logic*. Springer-Verlag, 1992.

[MV98] Fabien Morel and Vladimir Voevodsky. "$A^1$-homotopy theory of schemes". In: (1998).

[NPS01] B Nordström, K Petersson, and JM Smith. *Martin-Löf's type theory, Handbook of logic in computer science: Volume 5: Logic and algebraic methods*. Oxford University Press, Oxford, 2001.

[Str93] Thomas Streicher. "Investigations Into Intensional Type Theory". Habilitationsschrift. LMU München, 1993.

[WX10] Guozhen Wang and Zhouli Xu. "A Survey of Computations of Homotopy Groupes of Spheres and Cobordisms". In: (2010).