Modalities and Étale Maps in Homotopy Type Theory

Student Number: 1042317

Word Count: 7386
produced by `texcount`

University of Oxford

A dissertation submitted for the degree of

Master of Mathematics and Computer Science

Trinity 2023

Contents

1 Homotopy Type Theory **3**

1.1 Martin-Löf Type Theory . . . 3

1.1.1 Dependent function types . . . 4

1.1.2 Dependent Pair Types . . . 5

1.1.3 Coproduct Types . . . 6

1.1.4 Natural Numbers . . . 6

1.1.5 Unit and Empty Types . . . 6

1.1.6 Identity Types . . . 6

1.1.7 Universes . . . 8

1.1.8 Propositions as Types . . . 8

1.2 Homotopy Type Theory . . . 9

1.2.1 The Homotopy Interpretation . . . 9

1.2.2 Truncation Levels . . . 10

1.2.3 Fibres and Equivalences . . . 11

1.2.4 Univalence . . . 12

1.2.5 Transport and groupoid laws . . . 13

1.2.6 Higher Inductive Types . . . 14

1.2.7 Homotopy Pullbacks . . . 16

2 Modalities **18**

2.1 Reflective Subuniverses . . . 18

2.2 Formal Geometry . . . 21

2.3 Étale maps . . . 23

2.4 Orthogonal Factorisation Systems . . . 25

2.5 Étale Maps are Geometric, Again . . . 27

3 Nullification **31**

3.1 Constructing Nullification 31

3.2 Properties 34

3.3 Étale maps for nullification 36

Bibliography **41**

Introduction

Homotopy type theory (HoTT), or univalent foundations, has seen major interest in the past decade since its conception by Voevodsky [Voe06]. HoTT is a new, higher dimensional, foundational system, with its "types" acting more like homotopy types of topological spaces or ∞-groupoids than the sets of traditional set theoretic foundations. This offers benefits for higher dimensional category theorists, allowing natural constructions of previously complicated structures such as ∞-groups [BDR18; BR21].

This work focuses on a specific higher categorical structure: Modalities. Modalities are operators on higher categories which encode nice subcategories that often carry interesting geometric content. Modalities have seen important applications to a novel presentation of mathematical physics described by Urs Schreiber in the language of higher category theory [KS17; Sch13]. Interest has developed in translating Schreiber's work into HoTT, potentially simplfying its presentation. This has been started by Shulman [Shu15], Cherubini [Che18] and Myers [Mye22b; Mye22a]. The reformulation of Schreiber's work relies on a robust understanding of the behaviour of modalities in HoTT.

In the first chapter we introduce type theory, HoTT and its basic properties. Our main references are "Homotopy Type Theory" [Uni13] and "Introduction to Homotopy Type Theory" [Rij19]. We continue by developing the theory of modalities in the language of HoTT, following "Modalities in Homotopy Type Theory" [SSR20]. We describe formal analogues of geometric structures such as tangent bundles, derivative maps and étale maps with respect to a given modality, first introduced into HoTT by Cherubini [Che18]. The étale maps are of great interest to us, and we show how their relatively straightforward definition allows the derivation of many expected geometric results. In the third chapter we describe nullification, an important class of modalities. We work towards progressing an open question, classifying the étale maps of nullification:

Quesiton 0.1. *Given a pointed type (D, d), are the étale maps for nullification precisely those that lift uniquely against the inclusion of the basepoint $\mathbf{1} \to D$?*

We streamline and generalise an existing presentation of the special case when D is the n-sphere. Almost all of the proof of this works the same when $D = \Sigma B$ is the suspension of another type, exluding one technical property about B, which we call goodness. We have not shown if types other than the n-sphere are good, however we hope this generalisation provides a framework for extending the result.

Chapter 1

Homotopy Type Theory

1.1 Martin-Löf Type Theory

We begin by describing Martin-Löf Type Theory (MLTT), which serves as the foundation of HoTT. The fundamental building blocks of MLTT are types and terms of those types. These are analagous to sets and elements in set theory, although the correspondence is not exact. For example, we might have a type $\mathbb{N}$ of natural numbers and a term 2 of type $\mathbb{N}$, which we donote $2 : \mathbb{N}$. A statement of this form $2 : \mathbb{N}$ is an example of a **judgement**. A judgement is not a proposition, or a question, about the number 2, as in set theory, but instead describes the construction of the term 2. Every term in MLTT comes with a type equipped. MLTT has four judgements:

1. A type — Asserts A is a type.
2. $A \equiv B$ — Asserts A and B are **judgementally equal** types.
3. $a : A$ — Asserts a is a term of type A.
4. $a \equiv b : A$ — Asserts a and b are **judgementally equal** terms of type A.

We assert that judgemental equality of types and terms form equivalence relations. In a formal presentation of MLTT several other rules would be required, however for our purposes these are irrelevant.

We are allowed to make judgements in the context of a collection of other typing judgements. A basic instance of this is writing "$x : A \vdash B(x)$ type". This means that the type $B(x)$ varies with the value of $x : A$. We call $B(x)$ a **type family** over A. For example we might have a judgement $n : \mathbb{N} \vdash \mathbb{R}^n$ type defining the type family of n-dimensional Euclidean space. Similarly we may have contextual judgements of the form $x : A \vdash \Phi : B(x)$. Here Φ is a formula depending on the variable x, giving a term of type $B(x)$.

MLTT has a collection of basic types and type formers that will enable mathematics to be done. Unlike sets, which are simply described by their elements, type constructions require several components in the form of a collection of syntactic rules. These are:

1. **Formation rules**: Describe how to build our new type out of other types.
2. **Introduction rules**: Describe how to form canonical elements of the type.
3. **Elimination rules**: Tell us how to define maps out of the type.
4. **Computation rules**: Describe how the introduction and elimination rules interact.

A key difference to set theory is we do not a priori know all the terms of a given type after defining it, only the canonical ones. For many types we may characterise their elements, using the internal notion of equality defined later.

1.1.1 Dependent function types

Dependent functions behave like normal functions but with varying codomains.

Definition 1.1. Let A be a type and $x : A \vdash B(x)$ be a type family.

1. Formation: There is a type $\prod_{x:A} B(x)$ of **dependent functions**, otherwise known as the **Π-type**. For notational convenience we will sometimes write this $(x : A) \to B(x)$.
2. Introduction: To form a dependent function, we need a contextual judgement of the form $x : A \vdash \Phi : B(x)$. Given such a formula we may obtain a term:

$$\lambda x.\Phi : \prod_{x:A} B(x)$$

   In more familiar notation we might write this as the function $x \mapsto \Phi$.
3. Elimination: For Π-types the elimination rule enables us to apply a dependent functions to terms. Given any $f : \prod_{x:A} B(x)$ and $a : A$ we get a term $f(a) : B(a)$.
4. Computation: We have two rules called the β and η rules. The β-rule tells us how to evaluate dependent function. Given a judgement $x : A \vdash \Phi : B(x)$ for each $a : A$ we have:

$$(\lambda x.\Phi)(a) \equiv \Phi[a/x] : B(a)$$

Here $\Phi[a/x]$ is Φ with every occurrence of the variable x subsituted for a. The η-rule says for any $f : \prod_{x:A} B(x)$ we have:

$$\lambda x.f(x) \equiv f : \prod_{x:A} B(x)$$

This implies every element of $\prod_{x:A} B(x)$ is judgementally equal to a λ-expression.

Ordinary functions are the special case dependent functions over a constant type family $B(x) \equiv C$ for all $x : A$. In this case we denote $\prod_{x:A} B(x)$ as $A \to C$ or occasionally C^A. We take $\to$ to be right associative, so that $A \to B \to C$ is $A \to (B \to C)$.

From now on we will present the rules specifying a type more informally.

1.1.2 Dependent Pair Types

Similarly to Π-types, **dependent pair types**, or **Σ-types**, are a generalisation of cartesian products, where the type of the second value depends on the first. Given a type A and a type family $B(x)$ over A, we can form the type $\sum_{x:A} B(x)$. We sometimes use the more readable syntax $(x : A) \times B(x)$. The usual cartesian product is recovered when the $B(x)$ is constant. If $B(x) \equiv C$ for all $x : A$ we write $A \times C$ for $\sum_{x:A} B(x)$.

Given $a : A$ and $b : B(a)$ we can form a pair $(a, b) : \sum_{x:A} B(x)$.

The elimination rule says to define a dependent function from a Σ-type, it is enough to define it on the canonical elements: The pairs. Given a type family $D(z)$ for $z : \sum_{a:A} B(a)$ we can upgrade a dependent function

$$f : (a : A) \to (b : B(a)) \to D(a, b)$$

to a function

$$g : (p : \sum_{x:A} B(x)) \to D(p)$$

satisfying $g((a, b)) \equiv f(a, b)$ for $a : A$ and $b : B(a)$.

Example 1.2. Let $x : A \vdash B(x)$ be a type family. Using the elimination rule we may define projection maps soley on the pairs (a, b).

$$\mathtt{pr}_1 : \sum_{x:A} B(x) \to A \qquad \mathtt{pr}_2 : \left(z : \sum_{x:A} B(x) \right) \to B(\mathtt{pr}_1(z))$$
$$\mathtt{pr}_1(a, b) :\equiv a \qquad \mathtt{pr}_2(a, b) :\equiv b$$

1.1.3 Coproduct Types

Given types A and B we can form their coproduct $A + B$. This comes with maps $\mathtt{inl} : A \to A + B$ and $\mathtt{inr} : B \to A + B$ giving canonical elements. Suppose $C(z)$ is a type family over $z : A + B$. Given two functions $f : (a : A) \to C(\mathtt{inl}(a))$ and $g : (b : B) \to C(\mathtt{inr}(b))$ we can form their sum, $f + g : (z : A + B) \to C(z)$. This computes as $(f + g)(\mathtt{inl}(a)) \equiv f(a)$ and $(f + g)(\mathtt{inr}(b)) \equiv g(b)$.

1.1.4 Natural Numbers

We define the natural numbers $\mathbb{N}$ to be the type with canonical terms given by $0 : \mathbb{N}$ and a map $\mathtt{succ} : \mathbb{N} \to \mathbb{N}$. The elimination rule for the naturals is mathematical induction: Given a type family $B(n)$ over $\mathbb{N}$, an element $b_0 : B(0)$, and a function $g : \prod_{n:\mathbb{N}} B(n) \to B(\mathtt{succ}(n))$, we obtain a function $f : \prod_{n:\mathbb{N}} B(n)$. This comes with predictable computation rules:

$$f(0) \equiv b_0$$

$$f(\mathtt{succ}(n)) \equiv g(n, f(n))$$

We will denote $1 :\equiv \mathtt{succ}(0)$, $2 :\equiv \mathtt{succ}(1)$ and so on.

1.1.5 Unit and Empty Types

We assert the existence of a type $\mathbf{1}$ with a single contructor $\star : \mathbf{1}$. To define a dependent function $f : \prod_{x:\mathbf{1}} B(x)$ it suffices to give an element $b : B(\star)$. This map satisfies $f(\star) \equiv b$.

Similarly, we have a type $\mathbf{0}$ with no constructors. This automatically gives a (necessarily unique) dependent function $!_{\mathbf{0}} : \prod_{x:\mathbf{0}} B(x)$ for any type family over $\mathbf{0}$, making $\mathbf{0}$ the initial type. We interpret a function $A \to \mathbf{0}$ as a refuation that A is inhabited, since if it were, every type would be inhabited by initiallity. We thus define the negation of A to be $\neg A :\equiv A \to \mathbf{0}$.

1.1.6 Identity Types

The types we are defining are inductive. That is, they can be seen as the smallest type freely generated by the given constructors. For example the natural numbers are the free type with an element and an endomap. The identity types are defined as an inductive family of types. Given a type A for each $a, b : A$ we have a type $a =_A b$. This is the type of proofs that a and b are equal. For each $a : A$ we have a

constructor $\mathrm{refl}_a : a = a$ representing the reflexive proof that a equals a. This satisfies the following induction principle, often called the J-rule or **path induction**:

Given a type family $D(a, b, p)$ ranging over $a, b : A$ and $p : a =_A b$, and a function $g : \prod_{a:A} D(a, a, \mathrm{refl}_a)$ we obtain a function

$$f : \prod_{a,b:A} \prod_{p:a=_A b} D(a, b, p)$$

which computes as

$$f(a, a, \mathrm{refl}_a) \equiv g(a)$$

The identity types are preferred over judgemental equality since the latter only identifies syntactically equivalent terms. For instance given a variable $n : \mathbb{N}$ it is not true that $n + 1 \equiv 1 + n$, since these are syntactic terms with no way to reduce one to the other. However the type $n + 1 = 1 + n$ is inhabited.

Path induction is what allows for all the groupoidal structure of identites to be derived. As a first example we derive the concatenation and inverse operations on identites, corresponding to symmetry and transitivty of equality.

Lemma 1.3 (Groupoid structure on identities)**.** *Given a type A with $a, b, c : A$. Then we have:*

1. *A function $(-)^{-1} : a =_A b \to b =_A a$*
2. *A function $(- \cdot -) : a =_A b \to b =_A c \to a =_A c$*

Proof. 1. We are looking for a function $(-)^{-1} : \prod_{a,b:A} a =_A b \to b =_A a$. By path induction it is enough to define a function $\prod_{a:A} a =_A a$ which we can define by $\lambda a.\, \mathrm{refl}_a$. The function $(-)^{-1}$ then satisfies $\mathrm{refl}_a^{-1} \equiv \mathrm{refl}_a$

2. We are looking for a function:

$$(- \cdot -) : \prod_{a,b,c:A} a =_A b \to b =_A c \to a =_A c$$

Applying path induction twice it is enough to define $\mathrm{refl}_a \cdot \mathrm{refl}_a :\equiv \mathrm{refl}_a$.

□

More applications of path induction show the groupoidal laws are satisfied:

Lemma 1.4 (Groupoid laws on identities)**.** *Given a type A with $a, b, c, d : A$, $p : a = b$, $q : b = c$ and $r : c = d$ we terms of the following identity types:*

1. $\text{refl}_a \cdot p = p \cdot \text{refl}_b = p$.

2. $p \cdot p^{-1} = \text{refl}_a$ *and* $p^{-1} \cdot p = \text{refl}_b$.

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

1.1.7 Universes

Universes are large collections of types, similar to Grothendieck universes in set theory, which are closed under the type forming operations $\prod$, $\sum$ and $=$ specified before. We assert that for each finite collection of types or type families there is a universe $\mathcal{U}$ containing all of them. Using universes is a common way to describe type familes. A type family on A is a function $B : A \to \mathcal{U}$ for some universe $\mathcal{U}$. We will not include technical details regarding universes and we will work with only a single universe $\mathcal{U}$ throughout this dissertation.

1.1.8 Propositions as Types

Unlike set theory, MLTT is its own deductive system. This means there is no secondary logical system attatched, instead logic is done within the type theory. To state and prove logical statements in MLTT, we build a type corresponding to the statement and show it is inhabited. For example, we would say $n : \mathbb{N}$ is even if the following type is inhabited:

$$\sum_{m:\mathbb{N}} 2m = n$$

An inhabitant of this type is a pair (m, p) where $m : \mathbb{N}$ and $p : 2m = n$, witnessing a proof that n is even.

The relationship between logic and type theory is as follows:

Type Theory	Propositional Logic
Type X	Proposition X
Term $x : X$	Proof of Proposition X
Coproduct $X + Y$	Strong Disjunction $X \vee Y$
Cartesian Product $X \times Y$	Conjuction $X \wedge Y$
Functions $X \to Y$	Implication $X \Rightarrow Y$
Type family $P : X \to U$	Predicate $P(x)$
Π-type $\prod_{x:X} P(x)$	Universal Quantification $\forall x : P(x)$
Σ-type $\sum_{x:X} P(x)$	Strong Existential Quantification $\exists x : P(x)$

Strong disjunction refers to the fact that not only do we know X or Y is true, but we are told which one of them is true. Similarly in a strong existential quantification we are given a choice of an $x : X$ such that $P(x)$, not just told there exists one. This distinction is not often made in set theory, but will be important for us, and we will introduce the type theoretic weak notions later.

1.2 Homotopy Type Theory

We now describe notions introduced by Hoffman and Streicher [HS98] and Voevodsky [Voe06], extending MLTT with homotopy theoretic constructions. These allow a topological interpretation of HoTT in the model category of simplicial sets [KL18]. More recently HoTT has been shown to be the internal language of $(\infty, 1)$-topoi [Shu19], a large class of higher categories. As a consequence, HoTT can be used to describe complicated higher categorical structures with relative simplicity.

1.2.1 The Homotopy Interpretation

The innovation of univalent foundations was to notice that identity types in MLTT are more complex than traditional equality: They act more like paths in a topological space. Continuing this analogy we obtain a table of conversions between MLTT and homotopy theory:

Type Theory	Homotopy Theory
Type X	Homotopy Type X
Term $x : X$	Point of X
Equality $p : x = y$	Path between x and y in X
Type Family $B : X \to \mathcal{U}$	Fibration over the space X $\mathtt{pr}_1 : \sum_{x:X} B(x) \to X$
Dependent Function $f : \prod_{x:X} B(x)$	Section of the fibration
Pointwise Equality of $f, g : X \to Y$ $\prod_{x:X} f(x) = g(x)$	Homotopy of functions $f \sim g$

Using this translational tool, we can develop several homotopy theoretic constructions.

1.2.2 Truncation Levels

The identity types in MLTT can be between any types. Thus we can form higher identity types between terms of identity types. For some types, iterating higher paths eventually stabilises. Given the circle in set theoretic topology $\mathbb{S}^1 \subseteq \mathbb{R}^2$, all maps $\mathbb{S}^n \to \mathbb{S}^1$ are homotopic for $n \geq 2$. The circle thus has no homotopy information higher than its first homotopy, that is, it is 1-truncated. We now internalize truncation levels into type theory.

We start by defining **contractible types** and **propositions**.

$$\mathtt{isContr}(A) :\equiv \sum_{c:A} \prod_{a:A} c = a$$

$$\mathtt{isProp}(A) :\equiv \prod_{x,y:A} x = y$$

These form the base of the sequence of truncation levels. `isContr` translates to contractibility in topology since it describes a centre of contraction $c : A$, and a homotopy from the identity on A to the constant map at c. It can be shown that all the higher equalities of a contractible type are themselves contractible. Reading logically $\mathtt{isContr}(A)$ says A is a singleton.

The second truncation level is comprised of propositions. In a proposition all elements are equal, however the type might be empty. Types that are propositions intuitively represent properties, since a property is thought of as being either true (inhabited) or false (empty). Again it can be shown that the higher equality types of a proposition are contractible. We call the universe of propositions $\mathtt{Prop} :\equiv \sum_{A:\mathcal{U}} \mathtt{isProp}(A)$. This enables a handy description of subtypes of a type X, as maps $P : X \to \mathtt{Prop}$. We say x is contained in P if we have a term of $P(x)$.

We continue to define the heirarchy of truncation levels:

$$\mathtt{isTrunc} : \mathbb{Z}^{\geq -2} \to \mathcal{U} \to \mathcal{U}$$

$$\mathtt{isTrunc}_{-2}(A) :\equiv \mathtt{isContr}(A)$$

$$\mathtt{isTrunc}_{-1}(A) :\equiv \mathtt{isProp}(A)$$

And for $n \geq 1$

$$\mathtt{isTrunc}_{n+1}(A) :\equiv \prod_{x,y:A} \mathtt{isTrunc}_n(x = y)$$

Not all types are n-truncated for some n. A conjectural example is the two sphere S^2.

1.2.3 Fibres and Equivalences

Definition 1.5. Given a map $f : A \to B$ and $b : B$ we can form the **fibre** of f at b. This encodes the notion of preimage at a point.

$$\mathrm{fib}_f(b) :\equiv \sum_{a:A} f(a) = b$$

Definition 1.6. A map $f : A \to B$ is an **equivalence** if we have a term of the following type:

$$\mathtt{isEquiv}(f) :\equiv \prod_{b:B} \mathtt{isContr}(\mathrm{fib}_f(b))$$

We say A and B are equivalent if we have a term of type:

$$A \simeq B :\equiv \sum_{f:A\to B} \mathtt{isEquiv}(f)$$

Remark 1.7. This definition of equivalence is internalised from the notion of weak equivalence of topological spaces.

A remarkable feature of equivalence is it encodes isomorphism on standard mathematical structures. For example, equivalence on the type of groups is group isomorphism, on metric spaces is isometry and on metrizable spaces is homeomorphism. A practical benefit of HoTT is unifying these different notions of isomorphism into one encompassing definition of equivalence.

Additionally two types are equivalent iff there are mutually inverse maps between them. We favour equivalences over inverses since the type of inverses of a map is not a proposition: it can have multiple distinct elements. The type of equivalences, however, is a proposition [Uni13, Lemma 4.4.4].

Example 1.8. The identity function $\mathtt{id} : A \to A$ is an equivalence since for all $b : A$ we have:

$$\begin{aligned}\mathrm{fib}_{\mathtt{id}}(b) &\equiv \sum_{a:A} \mathtt{id}(a) = b \\ &\equiv \sum_{a:A} a = b\end{aligned}$$

This type is contractible, with centre of contraction (b, refl_b) by path induction.

1.2.4 Univalence

As it is, the identity type of MLTT is underspecified. Additional axioms must be assumed to inhabit this with enough elements to successfully perform mathematics. Traditionally "Axiom K" was chosen, however this eliminates the possibility of higher equality type, by asserting for any $p, q : a = b$ we have $p = q$. The downside is the loss of the ∞-groupoidal nature of types. A solution to this is the univalence axiom, which additionally incorparates the common mathematical intuition that isomorphic objects are indistinguishable.

We note two equal types are always equivalent, since we can define a function $\texttt{idToEquiv} : \prod_{A,B:\mathcal{U}}(A = B) \to (A \simeq B)$ by path induction. On (A, A, refl_A), define this to be the identity equivalence from Example 1.8.

Axiom 1.9 (Univalence)**.** For any universe $\mathcal{U}$, the function `idToEquiv` is itself an equivalence.

A slogan (though as it turns out equivalent) form of Univalence is that $(A = B) \simeq (A \simeq B)$.

An important consequence of the univalence axiom is function extensionality, a result which says two maps are equal iff they are pointwise equal.

Definition 1.10. Using path induction, for all $f, g : A \to B$ we get a map

$$\texttt{funExt}^{-} : f = g \to \left(\prod_{x:A} f(x) = g(x) \right)$$

defined on refl_f by

$$\texttt{funExt}^{-}(\mathrm{refl}_f) :\equiv \lambda x.\, \mathrm{refl}_{f(x)}$$

Theorem 1.11 (Function Extensionality (Section 4.9 [Uni13]))**.** *For $f, g : A \to B$ the map* `funExt`$^{-}$ *is an equivalence. We call the inverse* `funExt`*.*

Once we have univalence, identity types can now have more than one element, and all homotopical constructions work as expected. For example we can define the loop spaces of a pointed type:

Definition 1.12. A **pointed type** is a pair (X, x) where X is a type and $x : X$ Given two pointed types (X, x) and (Y, y) the pointed maps are

$$X \to_* Y :\equiv (f : X \to Y) \times (f(x) = y)$$

Definition 1.13. Given a pointed type (X, x) its **loop space** is the pointed type $\Omega(X, x) :\equiv ((x = x), \mathrm{refl}_x)$.

More generally for each $n : \mathbb{N}$ the $n^{\mathbf{th}}$ **loop space** is defined inductively as

$$\Omega^0(X, x) :\equiv X$$

$$\Omega^{n+1}(X, x) :\equiv \Omega(\Omega^n(X, x))$$

1.2.5 Transport and groupoid laws

In HoTT we interpret types as ∞-groupoids. This means that all functions should be functors between these groupoids, which we show now.

Lemma 1.14. *Let $f : A \to B$ and $p : a_1 = a_2$ in A. Then we have a term $\mathtt{ap}_f(p) : f(a_1) = f(a_2)$*

Proof. This follows from path induction. Assume $a_1 \equiv a_2$ and $p \equiv \mathrm{refl}_{a_1}$. Then define $\mathtt{ap}_f(\mathrm{refl}_{a_1}) :\equiv \mathrm{refl}_{f(a_1)}$. □

Lemma 1.15 (Groupoid Laws)**.** *Let $f : A \to B$ and $g : B \to C$, with $p : a_1 =_A a_2$, $q : a_2 =_A a_3$. Then:*

1. $\mathtt{ap}_{g \circ f}(p) = (\mathtt{ap}_g \circ \mathtt{ap}_f)(p)$
2. $\mathtt{ap}_{\mathtt{id}}(p) = p$
3. $\mathtt{ap}_f(p \cdot q) = \mathtt{ap}_f(p) \cdot \mathtt{ap}_f(q)$
4. $\mathtt{ap}_f(p^{-1}) = ap_f(p)^{-1}$

Proof. All four of these follow immediately from path induction. □

All the higher coherences necessary for a functor between ∞-groupoids can also be shown.

We also have a notion known as transport. Given a type family $B : A \to \mathcal{U}$ and $p : a_1 =_A a_2$ what can we say about $B(a_1)$ and $B(a_2)$? We cannot directly compare elements since these are different types, however there is a natural function between them, called the **transport** of p over B.

Definition 1.16. Let $B : A \to \mathcal{U}$ and $p : a_1 =_A a_2$. Transporting along p over B is the function $p_* : B(a_1) \to B(a_2)$ defined by path induction as $(\mathrm{refl}_{a_1})_* = \mathtt{id}_{B(a_1)}$

1.2.6 Higher Inductive Types

So far the types we have introduced have been freely generated by a collection of constructors, defining canonical elements of our type. HoTT allows a more general notion of constructor, in which we no longer just specify elements, but also certain identities between canonical terms. Types specified in this way are known as **Higher Inductive Types** (HITs). HITs can be used to perform standard topological constructions in type theory.

Example 1.17. We may form the circle S^1 as the HIT with constructors:

$$* : S^1 \qquad \mathit{loop} : * = *$$

The circle has all expected homotopy theoretic properties. For example it can be shown that $(* = *) \simeq \mathbb{Z}$ [Uni13, Corollary 8.1.10] and hence that the fundamental group $\pi_1(S^1) \simeq \mathbb{Z}$.

This definition of the circle does not generalise to higher dimensions easily. A more principled definition uses **suspensions**.

Definition 1.18. Given a type X its **suspension** is the higher inductive type ΣX with the following constructors:

1. Points $N, S : \Sigma X$.
2. A function $\texttt{merid} : X \to (N = S)$.

We consider ΣX as a pointed type with point $N : \Sigma X$

The visual intuition for the suspension is a "sphere" with equator X. N and S are the north and south pole, and for each $x : X$ we have a meridian line $\texttt{merid}(x)$ from N to S "through x". For each point in X we have a different path from N to S. From this definition we have an induction principle, known as the universal property of suspensions. This tells us how to map out of a suspension:

$$(\Sigma A \to B) \simeq ((b_N : B) \times (b_S : B) \times (A \to b_N = b_S))$$

Definition 1.19. The n-sphere for $n : \mathbb{N}$ is defined inductively:

$$S^0 :\equiv \mathbf{2} :\equiv \mathbf{1} + \mathbf{1}$$

$$S^{n+1} :\equiv \Sigma S^n$$

The previous definition of the circle agrees with this one [Uni13, Lemma 6.5.1].

The main benefit of this definition is the ability to use loop space/suspension adjunction to completely characterise maps from the n-sphere.

Theorem 1.20 (Suspension/Loop Space Adjunction)**.** *For all pointed types* (A, a_0), (B, b_0) *we have*

$$(\Sigma A \to_* B) \simeq (A \to_* \Omega B)$$

Proof. We compute:

$$\begin{aligned}
\Sigma A \to_* B &\equiv (f : \Sigma A \to B) \times (f(N) = b_0) \\
&\simeq ((b_N : B) \times (b_S : B) \times (A \to b_N = b_S)) \times (b_N = b_0) \\
&\simeq (b_S : B) \times (A \to b_0 = b_S) \\
&\simeq (b_S : B) \times (f : A \to b_0 = b_S) \times (p : b_0 = b_S) \times (f(a_0) = p)
\end{aligned}$$

The first equivalence is given by the universal property, the second is given by concatenating with the proof $b_0 = b_N$, and the third is given by noting $(p : b_0 = b_S) \times (f(a_0) = p)$ is contractible onto centre $(f(a_0), \mathrm{refl}_{f(a_0)})$. Similarly $(b_S : B) \times (p : b_0 = b_S)$ contracts onto $(b_0, \mathrm{refl}_{b_0})$. This gives

$$\Sigma A \to_* B \simeq (f : A \to b_0 = b_0) \times (f(a_0) = \mathrm{refl}_{b_0}) \equiv A \to_* \Omega B$$

□

Corollary 1.21. *For all* $n : \mathbb{N}$ *we have* $S^n \to_* A \simeq \Omega^n A$

Proof. By repeated application of the adjunction we have $S^n \to_* A \simeq \mathbf{2} \to_* \Omega^n A \simeq \Omega^n A$. The last equivalence is since a pointed map from $\mathbf{2}$ simply picks out a single point. □

Another important higher inductive type is **propositional truncation**. This construction takes a type and squashes all higher homotopy information, leaving only a proposition.

Definition 1.22. For a type A its **propositional truncation** $\|A\|$ is the HIT with constructors:

$$|-| : A \to \|A\| \qquad (x, y : \|A\|) \to x = y$$

The second constructor tells us precisely that $\|A\|$ is a proposition. We think of $\|A\|$ as the proposition "A is inhabited". This satisfies the elimination rule that any proposition B and map $f : A \to B$ induces $\bar{f} : \|A\| \to B$ such that $\bar{f}(|a|) \equiv f(a)$.

Propositional truncation allows us to define the weak existential quantifier, mentioned earlier. Given a type A and a type family $B : A \to \mathcal{U}$ we say there weakly exists $a : A$ such that $B(a)$ if we have a term of type $\| \sum_{a:A} B(a) \|$. This type forgets the choice of a specific $a : A$ such that $B(a)$, and simply tells us there is one. Similarly we may construct the weak disjunction of types X and Y to be $\|X + Y\|$. This allows us to encode more faithfully ideas from set theory.

Definition 1.23. A map $f : A \to B$ is **surjective** if for all $b : B$ we have a term of type $\| \operatorname{fib}_f(b) \| \equiv \| \sum_{a:A} f(a) = b \|$.

1.2.7 Homotopy Pullbacks

A benefit of HoTT is the ease with which homotopy limits and colimits can be described. Since everything is automatically up to homotopy, simply interpreting the normal rules for limits and colimits gives the correct notions. For the geometric goals we have in this work, the language of pullbacks is extremely important. To this end we recall some of the theory of homotopy pullbacks now.

Definition 1.24. Let $f : A \to C$ and $g : B \to C$. Then define the **pullback** to be:

$$A \times_C B :\equiv \sum_{x:A} \sum_{x:B} (f(a) = g(b))$$

This has natural maps given by:

$$\begin{array}{ll} A \times_C B \to A & A \times_C B \to B \\ (a, b, p) \mapsto a & (a, b, p) \mapsto b \end{array}$$

Definition 1.25. Consider a commutative square:

$$\begin{array}{ccc} P & \xrightarrow{h} & A \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[g]{} & C \end{array}$$

That is we have an identity $p : f \circ h = g \circ k$. We say this is a **pullback square** if for any type X, the natural map

$$P^X \to A^X \times_{C^X} B^X$$

$$\phi \mapsto (h \circ \phi, k \circ \phi, \mathtt{ap}_{- \circ \phi}(p))$$

is an equivalence. Noting the data of a commutative square is equivalent to an element of the type $A^X \times_{C^X} (B^X)$, we may unpack this definition to the usual universal property of the pullback.

We now state results about pullbacks that will be useful for the rest of the work.

Lemma 1.26 ([Rij19] Proposition 1.4.7)**.** *For any pair $f : A \to C$ and $g : B \to C$, the following square is a pullback square:*

$$\begin{array}{ccc} A \times_C B & \longrightarrow & A \\ \downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[g]{} & C \end{array}$$

Lemma 1.27 ([Rij19] Corollary 1.4.13)**.** *A commuting square*

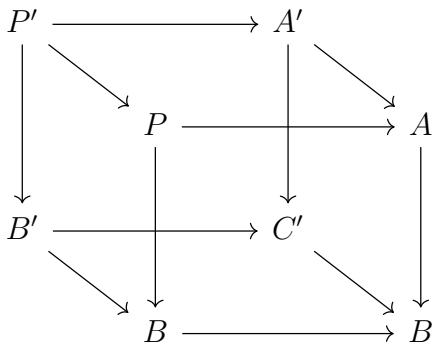
$$\begin{array}{ccc} P & \xrightarrow{h} & A \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[g]{} & C \end{array}$$

with $p : f \circ h = g \circ k$ is a pullback iff for all $b : B$ the natural map

$$\begin{aligned} \mathrm{fib}_k(b) &\to \mathrm{fib}_f(g(b)) \\ (x, \gamma) &\mapsto (h(x), \mathtt{funExt}^{-}(p, b) \cdot ap_g(\gamma)) \end{aligned}$$

is an equivalence.

Lemma 1.28 ([Rij19] Corollary 2.1.18)**.** *Suppose we have a commuting cube*



in which the front and left squares are pullbacks. Then the back square is a pullback iff the right square is.

Chapter 2

Modalities

Modalities have their origins in philosophy and logic. In this setting modalities are operations on propositions, modifying them to represent altered statements. Traditional examples include asking if a proposition is "possibily" true or "neccessarily" true. Another example is from temporal logic, a field with many applications in formal verification, where we might consider if a statement is "eventually" true.

In type theory modalities play a significant role. Homotopy type theory is the internal language of all $(\infty, 1)$-topoi [Shu19]. An important part of the theory of $(\infty, 1)$-topoi is describing nice subcategories, in particular reflective subcategories. These subcategories enable descriptions of various geometric structures. Introducing modalities can involve changing the syntax of type theory. For example, Shulman introduced a system of modalities to form Cohesive HoTT [Shu15] which internalised the theory of cohesive $(\infty, 1)$-topoi [Sch13] into type theory. More general modal type theories, such as Multi-Mode Dependent Type theory allow the introduction of large classes of modalities [Gra+21]. However, the language of HoTT is already powerful enough to represent certain classes of modalities. In particular idempotent, monadic modalities are straightforward to define. We focus on building the theory of these modalities, and the geometric structures they encode.

2.1 Reflective Subuniverses

Definition 2.1. A **reflective subuniverse** of $\mathcal{U}$ consists of:

- A family $\texttt{isModal} : \mathcal{U} \to \texttt{Prop}$.
- A **modal operator** $\bigcirc : \mathcal{U} \to \mathcal{U}$.
- A **modal unit** $\eta : \prod_{A:\mathcal{U}} A \to \bigcirc A$

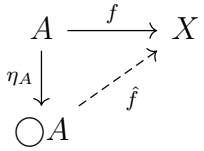
These must satisfy:

- For all $A : \mathcal{U}$ we have $\texttt{isModal}(\bigcirc A)$.
- **Modal recursion**: For all $B : \mathcal{U}$ satisfying $\texttt{isModal}(B)$ the function

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

  is an equivalence.

Remark 2.2. The second condition above is a conceptually satisfying way of encoding a higher coherent universal property with all higher coherences in HoTT. Expanding the definition of equivalence, we may restate modal recursion: for all $f : A \to B$ where B is modal, we have a unique map $\hat{f} : \bigcirc A \to B$ such that $f = \hat{f} \circ \eta_A$.



By uniqueness we mean that the type of functions making the diagram commute is contractible.

Lemma 2.3. *Reflective subuniverses automatically preserves many type formers. In particular:*

1. ***Stability under*** $\prod$*: If* $A : \mathcal{U}$ *and* $B : A \to \mathcal{U}$ *satisfies* $\texttt{isModal}(B(a))$ *for all* $a : A$ *then* $\texttt{isModal}(\prod_{a:A} B(a))$.
2. ***Stability under*** $=$*: Given* $A : \mathcal{U}$ *with* $\texttt{isModal}(A)$*, and* $x, y : A$ *we have* $\texttt{isModal}(x = y)$.

Proof. For (1) see [Uni13], Theorem 7.7.2. For (2) see [SSR20], Lemma 1.25. □

However $\sum$-types are not necessarily preserved. By enforcing that these be preserved too we gain an extremely useful property: modal recursion generalises to dependent types, giving a modal induction principle.

Definition 2.4. A reflective subuniverse is **Σ-closed** if for $A : \mathcal{U}$ and $B : A \to \mathcal{U}$ satisfying $\texttt{isModal}(A)$ and $\texttt{isModal}(B(a))$ for all $a : A$ we have $\texttt{isModal}(\sum_{a:A} B(a))$.

We will refer to Σ-closed reflective subuniverses as **modalities**. From now on fix a modality given by $\bigcirc$, η and $\texttt{isModal}$.

Proposition 2.5 (Modal Induction [Uni13], Theorem 7.7.4)**.** *Given $B : \bigcirc A \to \mathcal{U}$ such that for all $z : \bigcirc A$ we have* $\texttt{isModal}(B(z))$*, the precomposition map*

$$\prod_{z:\bigcirc A} B(z) \to \prod_{a:A} B(\eta_A(a))$$

is an equivalence. We call the inverse map $\texttt{Ind}_{\bigcirc}$*.*

When we have a modality, the data of `isModal` can be recovered from the unit η.

Lemma 2.6. *We have* $\texttt{isModal}(A)$ *iff $\eta_A : A \to \bigcirc A$ is an equivalence.*

Proof. Suppose $\texttt{isModal}(A)$. Define $g : \bigcirc A \to A$ by induction on the identity map on A. Then $g \circ \eta_A = \texttt{id}_A$. Since for $a : A$ we have $(\eta_A \circ g)(\eta_A(a)) = \eta_A(a)$, we may apply modal induction to deduce $\eta_A(g(z)) = z$ for all $z : \bigcirc A$. This is valid since equality $\bigcirc A$ is modal by Lemma 2.3. Then by function extensionality $\eta_A \circ g = \texttt{id}$ Hence g is an inverse to η_A and so η_A is an equivalence.

The converse holds since if η_A is an equivalence then A is equivalent to the modal type $\bigcirc A$ and thus is modal. □

Corollary 2.7. *Any contractible type is modal.*

Proof. Suppose A is contractible. Applying modal induction and contractibility we deduce for all $x, y : \bigcirc A$ that $x = y$. Since A is pointed, so is $\bigcirc A$, and we deduce $\bigcirc A$ is contractible. Then η_A is an equivalence, since any map between two contractible types is, and A is modal. □

Lemma 2.8. *Given $A, B : \mathcal{U}$ with* $\texttt{isModal}(A)$ *and* $\texttt{isModal}(B)$ *we have* $\texttt{isModal}(A = B)$*.*

Proof. By univalence

$$A = B \simeq (A \simeq B)$$

Unpacking the definition of $A \simeq B$, and applying Σ-closedness and stability under type formers in Lemma 2.3, we deduce $A \simeq B$ is modal. Hence $A = B$ is modal. □

Example 2.9. There are several examples of modalities appearing naturally in HoTT:

1. **Propositional truncation**: The truncation $\| - \|$ acts as the modal operator, $| - |$ as the modal unit. The modal types are the propositions.

2. **$\neg\neg$-modality**: Define the modal operator as $A \mapsto \neg\neg A$. The modal types are the $\neg\neg$-stable propositions. The unit is $A \to \neg\neg A$ given by

$$a : A \mapsto (\phi : (A \to \mathbf{0}) \mapsto \phi(a))$$

Lemma 2.10 ([SSR20] Lemma 1.24)**.** *For any X and $P : X \to \mathcal{U}$ we have*

$$\bigcirc \sum_{x:X} P(x) \simeq \bigcirc \sum_{x:X} (\bigcirc P(x))$$

Lemma 2.11 (Functoriality and naturality of $\bigcirc$)**.** *Given $f : A \to B$ we have a unique map $\bigcirc f : \bigcirc A \to \bigcirc B$ such that the following naturality diagram commutes:*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle \eta_A}\downarrow & & \downarrow{\scriptstyle \eta_B} \\ \bigcirc A & \xrightarrow[\bigcirc f]{} & \bigcirc B \end{array}$$

Further, the assignment of this map is functorial.

Proof. Let $f : A \to B$. Since $\eta_B \circ f : A \to \bigcirc B$ and $\bigcirc B$ is modal we obtain a unique map $\bigcirc f : \bigcirc A \to \bigcirc B$ such that $\eta_B \circ f = \bigcirc f \circ \eta_A$ by modal recursion. So the naturality diagram commutes.

This preserves identity maps since the naturailty square commutes with either choice of map $\bigcirc A \to \bigcirc A$:

$$\begin{array}{ccc} A & \xrightarrow{\mathtt{id}_a} & A \\ {\scriptstyle \eta_A}\downarrow & & \downarrow{\scriptstyle \eta_A} \\ \bigcirc A & \underset{\mathtt{id}_{\bigcirc A}}{\overset{\bigcirc \mathtt{id}_a}{\rightrightarrows}} & \bigcirc A \end{array}$$

By uniqueness of $\bigcirc \mathtt{id}_a$ we deduce $\bigcirc \mathtt{id}_a = \mathtt{id}_{\bigcirc A}$.

Similarly, the outer square diagram commutes for any $f : A \to B$ and $g : B \to C$:

$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{g} & C \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ \bigcirc A & \xrightarrow{\bigcirc f} & \bigcirc B & \xrightarrow{\bigcirc g} & \bigcirc C \\ & \searrow & & \nearrow & \\ & & \bigcirc(g \circ f) & & \end{array}$$

By uniqueness $\bigcirc g \circ \bigcirc f = \bigcirc(g \circ f)$. □

2.2 Formal Geometry

The use of arbitrary modalities to perform geometric constructions was introduced by Khavkine and Schreiber [KS17]. Cherubini partially translated this work into HoTT [Che18]. We review some of these definitions.

Definition 2.12. Let $A : \mathcal{U}$ and $a : A$. The $\bigcirc$**-disk** around a is the type:

$$\mathbb{D}^{\bigcirc}(A, a) :\equiv \sum_{x:A} \eta_A(a) = \eta_A(x)$$

Equivalently $\mathbb{D}^{\bigcirc}(A, a)$ is the following pullback:

$$\begin{array}{ccc} \mathbb{D}^{\bigcirc}(A, a) & \longrightarrow & \mathbf{1} \\ \downarrow & & \downarrow{\scriptstyle \mathtt{const}_{\eta_A(a)}} \\ A & \xrightarrow{\eta_A} & \bigcirc A \end{array}$$

This definition has pre-existing geometric intuition from synthetic differential geometry, where Penon defined the infinitesimal neighbourhood of a point x as the set of points y such that $\neg\neg(x = y)$ [Pen81], that is the $\neg\neg$-disk around x. Following this example, we may imagine the $\bigcirc$-disk as an infinitesimal disk around a point. In this way they behave similary to tangent spaces.

Lemma 2.13. *For $A, B : \mathcal{U}$ and $f : A \to B$, if $p : \eta_A(a) = \eta_A(b)$ then we have a term:*

$$f_*(p) : \eta_B(f(a)) = \eta_B(f(b))$$

Proof. We calculate:

$$\begin{aligned} \eta_B(f(a)) &= \bigcirc f(\eta_A(a)) && \text{By naturality of } \bigcirc f \\ &= \bigcirc f(\eta_A(b)) && \text{By } \mathtt{ap}_{\bigcirc f}(p) \\ &= \eta_B(f(b)) && \text{By naturality of } \bigcirc f \end{aligned}$$

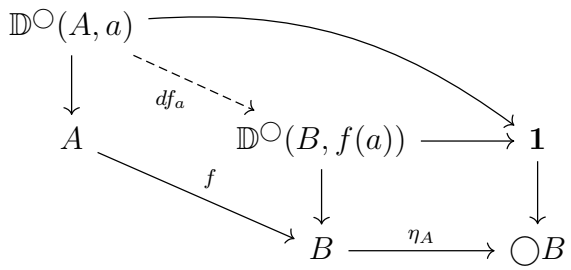
□

Definition 2.14. Let $A, B : \mathcal{U}$ and $f : A \to B$. Lemma 2.13 allows us to define the **differential** of f at $a : A$:

$$\begin{aligned} df_a : \mathbb{D}^{\bigcirc}(A, a) &\to \mathbb{D}^{\bigcirc}(B, f(a)) \\ df_a(x, p) &:\equiv (f(x), f_*(p)) \end{aligned}$$

We can equivalently define the differential using the universal property of pullbacks.

Noting the outer square commutes by Lemma 2.13, df_a is the unique map making



commute.

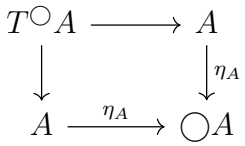
Collecting all $\bigcirc$-disks together we can also form an infinitesimal analogue of the tangent bundle.

Definition 2.15. The **$\bigcirc$-tangent bundle** of A is the type

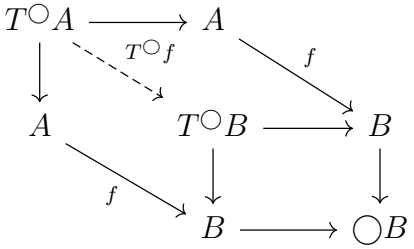
$$T^{\bigcirc}A :\equiv \sum_{a:A} \mathbb{D}^{\bigcirc}(A,a)$$

Given $f : A \to B$ there is an induced map $T^{\bigcirc}f : T^{\bigcirc}A \to T^{\bigcirc}B$ given by $f(a,v) = (f(a), df_a(v))$

Again the tangent bundle is a pullback:



We can describe the induced map using the universal property of the pullback:



(2.1)

2.3 Étale maps

We now define étale maps. Étale maps in type theory correspond both to formally étale maps in algebraic geometry and local diffeomorphisms in differential geometry.

Definition 2.16. A map $f : X \to Y$ is **◯-étale** if the following square is a pullback:

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & \bigcirc X \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \bigcirc f} \\ Y & \xrightarrow[\eta_Y]{} & \bigcirc Y \end{array}$$

Given this definition alone, the similarity to other geometric notions is not clear. We will describe two ways of illustrating this connection.

The first method relates ◯-étale maps to differential geometry. In differential geometry a map of smooth manifolds $f : X \to Y$ is a local diffeomorphism iff the diagram

$$\begin{array}{ccc} TX & \xrightarrow{Tf} & TY \\ \downarrow & & \downarrow \\ X & \xrightarrow{f} & Y \end{array}$$

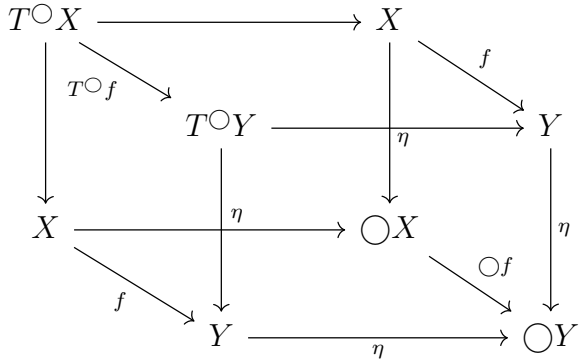
is a pullback [nLa]. Here TX and TY are the actual tangent bundles of the manifolds. For ◯-étale maps we have the following.

Lemma 2.17. *If $f : X \to Y$ is ◯-étale then*

$$\begin{array}{ccc} T^{\bigcirc}X & \xrightarrow{T^{\bigcirc}f} & T^{\bigcirc}Y \\ \downarrow{\scriptstyle \mathtt{pr}_1} & & \downarrow{\scriptstyle \mathtt{pr}_1} \\ X & \xrightarrow{f} & Y \end{array}$$

is a pullback.

Proof. Adding the missing faces of the cube from Equation 2.1 we obtain a commutative cube.



Since f is ◯-étale, the right and bottom faces are pullbacks. By definition of the tangent bundle the front and back faces are pullbacks. Thus by rotations of Lemma 1.28 the two other faces of the cube are pullbacks, giving the result. □

Another result of traditional differential geometry is that local diffeomorphisms induce isomorphisms on induced maps on tangent spaces [nLa]. The corresponding statement in type theory, that the differential maps between formal disks are isomorphisms, follows immediately from the previous lemma. This is because the square

$$\begin{array}{ccc} T^{\bigcirc}X & \xrightarrow{T^{\bigcirc}f} & T^{\bigcirc}Y \\ \downarrow & & \downarrow \\ X & \xrightarrow{f} & Y \end{array}$$

is a pullback iff the induced maps on fibres are equivalences, by Lemma 1.27. It is quick to see these induced maps are precisely the derivative maps.

We may hope the converse to Lemma 2.17 holds in type theory, as it does in differential geometry. However, for this we require additional assumptions about the modality. Specifically, we need the map $\eta_X : X \to \bigcirc X$ to be surjective. Luckily, for more geometric modalities this condition is always satisfied, as will be discussed in Chapter 3.

Lemma 2.18. *Suppose $f : X \to Y$ is such that*

$$\begin{array}{ccc} T^{\bigcirc}X & \xrightarrow{T^{\bigcirc}f} & T^{\bigcirc}Y \\ \downarrow & & \downarrow \\ X & \xrightarrow{f} & Y \end{array}$$

is a pullback. Suppose further the map $\eta_X : X \to \bigcirc X$ is surjective. Then f is $\bigcirc$-étale.

Proof. By Lemma 1.27 it is enough to show for all $z : \bigcirc X$ the induced map on fibres $\phi : \mathrm{fib}_{\eta_X}(z) \to \mathrm{fib}_{\eta_Y}(\bigcirc f(z))$ is an equivalence. Since η_X is surjective we have $\|z = \eta_X(x)\|$ for some $x : X$. Since we are trying to prove $\mathtt{isEquiv}(\phi)$, which is a proposition, we can obtain a term $p : z = \eta_X(x)$, by the elimination rule of propositional truncation. Then ϕ is equivalent via p to the map $df_x : \mathbb{D}^{\bigcirc}(X, x) \to \mathbb{D}^{\bigcirc}(Y, f(x))$. By assumption that the tangent square is a pullback we know df_x is an equivalence, and so is ϕ. □

2.4 Orthogonal Factorisation Systems

The promised second method of relating $\bigcirc$-étale maps to more traditional geoemtric notions requires the development of some machinery. We will show that $\bigcirc$-étale maps satisfy a lifting condition. This is analogous to scheme theory in which formally étale maps are those that lift against first order thickenings of schemes [Sta23, Tag 02HF].

In analogy with this, $\bigcirc$-étale maps lift against maps inducing an equivalence on the modal level. To develop a suitably nice theory of lifting conditions, we introduce orthogonal factorisation systems.

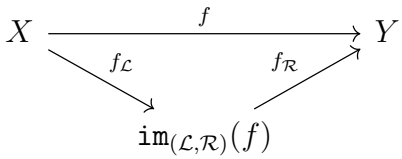
Definition 2.19. An **orthogonal factorisation system** is formed of two collections of maps $\mathcal{L}, \mathcal{R} : \prod_{X,Y:\mathcal{U}}(X \to Y) \to \mathtt{Prop}$ sastisfying:

1. $\mathcal{L}$ and $\mathcal{R}$ contain all equivalences, and are closed under composition.
2. For any map $f : X \to Y$ there is a type $\mathtt{im}_{(\mathcal{L},\mathcal{R})}(f)$ and maps

$$f_{\mathcal{L}} : X \to \mathtt{im}_{(\mathcal{L},\mathcal{R})}(f)$$

$$f_{\mathcal{R}} : \mathtt{im}_{(\mathcal{L},\mathcal{R})}(f) \to Y$$

   so that the following triangle commutes:



3. For all $i : A \to B$ in $\mathcal{L}$ and $f : X \to Y$ in $\mathcal{R}$ the following square is a pullback.

$$\begin{array}{ccc} X^B & \xrightarrow{i\circ\_} & X^A \\ {\scriptstyle \_\circ f}\downarrow & & \downarrow{\scriptstyle \_\circ f} \\ Y^B & \xrightarrow[i\circ\_]{} & Y^A \end{array}$$

   We say i is **left orthogonal** to f and f **right orthogonal** to i.

We now unpack this definition. The first condition ensures that $\mathcal{L}$ and $\mathcal{R}$ specify subcategories of $\mathcal{U}$, containing all equivalences. The second condition comprises the "factorisation" element. Every map decomposes into a map in $\mathcal{L}$ followed by a map in $\mathcal{R}$.

Orthogonality is the most interesting component of the definition. It is best understood as encoding a lifting condition, which is often the setting orthogonality appears in traditionally. The square as given is a pullback iff the natural map $X^B \to X^A \times_{Y^A} Y^B$ is an equivalence. That is, for each element of $X^A \times_{Y^A} Y^B$ there is a unique element in X^B mapping onto it. An element of $X^A \times_{Y^A} Y^B$ is a triple

(g, h, p) where $g : A \to X$, $h : B \to Y$ and $p : g \circ f = i \circ h$. So $X^A \times_{Y^A} Y^B$ is the type of commutative squares of the form:

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

The natural map then sends a map $l : B \to X$ to the commutative square:

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

The square being a pullback tells us that for every commutative square in $X^A \times_{Y^A} Y^B$ there is a unique lift $B \to X$ making both triangles commute:

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

Furthermore the proofs of commutativity satisfy an extra coherence condition, but we will not detail this here as it does not matter for us.

Example 2.20. The most basic example of an orthogonal factorisation system is given by the classes of surjective and injective maps.

2.5 Étale Maps are Geometric, Again

Definition 2.21. A map $f : X \to Y$ is said to be an **$\bigcirc$-equivalence** if $\bigcirc f$ is an equivalence.

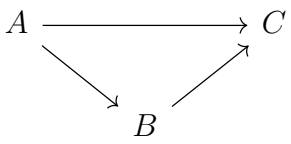
Example 2.22. We give some useful examples of $\bigcirc$-étale maps and $\bigcirc$-equivalences.

1. All modal units are $\bigcirc$-equivalences as $\bigcirc\eta : \bigcirc A \to \bigcirc\bigcirc A$ is an equivalence by Lemma 2.6, since $\bigcirc A$ is modal, and $\bigcirc\eta_A = \eta_{\bigcirc A}$ by induction.

2. For any $f : A \to B$, the map $\bigcirc f$ is $\bigcirc$-étale since in the following diagram the top and bottom maps are equivalences, making the diagram a pullback.

$$\begin{array}{ccc} \bigcirc A & \xrightarrow{\eta_{\bigcirc A}} & \bigcirc\bigcirc A \\ \downarrow{\scriptstyle \bigcirc f} & & \downarrow{\scriptstyle \bigcirc\bigcirc f} \\ \bigcirc B & \xrightarrow{\eta_{\bigcirc B}} & \bigcirc\bigcirc B \end{array}$$

Lemma 2.23 ([CR20] Proposition 6.3)**.** *$\bigcirc$-equivalences satisfy the 3-for-2 property. That is given a commuting triangle*



in which any two of the maps are $\bigcirc$-equivalences then the third is.

Lemma 2.24. *The classes of $\bigcirc$-étale maps and $\bigcirc$-equivalences:*

1. *Contain all equivalences*

2. *Stable under composition*

Proof. Equivalences are $\bigcirc$-étale maps since if f is an equivalence so is $\bigcirc f$ and hence the defining square is automatically a pullback. Compositions of étale maps are étale by pullback pasting.

Equivalences are $\bigcirc$-connected since their fibres are contractible, and thus modal, by Lemma 2.7. Equivalences are stable under composition by the 3-for-2 property. □

Lemma 2.25 ([CR20] Proposition 6.3)**.** *$\bigcirc$-equivalences are stable under pullback by $\bigcirc$-étale maps.*

Lemma 2.26. *A map $f : A \to B$ is an $\bigcirc$-equivalence iff for all $\bigcirc$-modal types X the post-composition map $f^X : X^B \to X^A$ is an equivalence.*

Proof. ($\Rightarrow$) Let X be a $\bigcirc$-modal type. Suppose $f : A \to B$ is an $\bigcirc$-equivalence. We have a chain of equivalences:

$$(B \to X) \simeq (\bigcirc B \to X) \simeq (\bigcirc A \to X) \simeq (A \to X)$$

$$g \mapsto \mathtt{Ind}_{\bigcirc}(g) \mapsto \mathtt{Ind}_{\bigcirc}(g) \circ \bigcirc f \mapsto \mathtt{Ind}_{\bigcirc}(g) \circ \bigcirc f \circ \eta_A$$

The first and last maps are equivalences by modal induction. The intermediate map is an equivalence as $\bigcirc f$ is. We calculate:

$$\begin{aligned} \mathtt{Ind}_{\bigcirc}(g) \circ \bigcirc f \circ \eta_A &= \mathtt{Ind}_{\bigcirc}(g) \circ \eta_B \circ f && \text{By naturality} \\ &= g \circ f && \text{By modal induction} \end{aligned}$$

So this composition of equivalences is composition with f, and the map $- \circ f = f^X$ is an equivalence.

($\Leftarrow$) Suppose $- \circ f : (B \to X) \to (A \to X)$ is an equivalence for all modal X. We extend this to an equivalence $(\bigcirc B \to X) \to (\bigcirc A \to X)$ using a similar chain of equivalences, giving the map $\mathtt{Ind}_\bigcirc \circ (- \circ f) \circ (- \circ \eta_B)$. We check this is map the composition $- \circ \bigcirc f$. By function extensionality it is enough to check for all $g : \bigcirc B \to X$ that $\mathtt{Ind}_\bigcirc(g \circ \eta_B \circ f) = g \circ \bigcirc f$. This follows from naturality, $\eta_B \circ f = \bigcirc f \circ \eta_A$, and the definition of $\mathtt{Ind}_\bigcirc$ as the inverse to $- \circ \eta_A$.

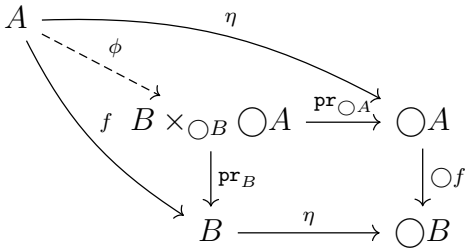
Now we define an inverse to $\bigcirc f$. Since the map

$$- \circ \bigcirc f : (\bigcirc B \to \bigcirc A) \to (\bigcirc A \to \bigcirc A)$$

is an equivalence there is $g : \bigcirc B \to \bigcirc A$ such that $g \circ \bigcirc f = \mathtt{id}_{\bigcirc A}$. Since g is left inverse to $\bigcirc f$ we have $\bigcirc f \circ g \circ \bigcirc f = \bigcirc f$. But composing on the right by $\bigcirc f$ is an equivalence, so $\bigcirc f \circ g = id_B$ and g is right inverse to $\bigcirc f$. So g is inverse to $\bigcirc f$ and $\bigcirc f$ is an equivalence. □

Theorem 2.27. *The classes of $\bigcirc$-equivalences and $\bigcirc$-étale maps from an orthogonal factorisation system.*

Proof. We already know that $\bigcirc$-equivalences and $\bigcirc$-étale maps are closed under composition and contain all equivalences. Let $f : A \to B$. Consider the diagram:

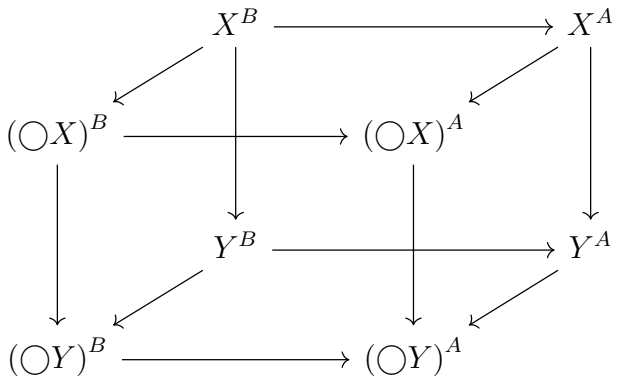


The universal property of pullbacks gives the map ϕ. $\mathtt{pr}_B$ is the pullback of the étale map $\bigcirc f$, and hence is étale. $\mathtt{pr}_{\bigcirc A}$ is the pullback of the $\bigcirc$-equivalence η, against étale maps $\mathtt{pr}_B$ and $\bigcirc f$. Hence by Lemma 2.25, $\mathtt{pr}_{\bigcirc A}$ is an $\bigcirc$-equivalence. By the 3-for-2 rule we deduce ϕ is an $\bigcirc$-equivalence. So we have factored $f = \mathtt{pr}_B \circ \phi$ as an $\bigcirc$-equivalence followed by an $\bigcirc$-étale map.

For orthogonality suppose $i : A \to B$ is an $\bigcirc$-equivalence and $f : X \to Y$ is an $\bigcirc$-étale map. We want to show that

$$\begin{array}{ccc} X^B & \xrightarrow{i \circ \_} & X^A \\ {\scriptstyle \_ \circ f}\downarrow & & \downarrow{\scriptstyle \_ \circ f} \\ Y^B & \xrightarrow[i \circ \_]{} & Y^A \end{array}$$

is a pullback. Consider the commutative cube:



Since $i : A \to B$ is an $\bigcirc$-equivalence the induced maps $(\bigcirc X)^B \to (\bigcirc X)^A$ and $(\bigcirc Y)^B \to (\bigcirc Y)^A$ are equivalences, by Lemma 2.26. So the front square of the cube is a pullback square. The square

$$\begin{array}{ccc} X & \longrightarrow & \bigcirc X \\ \downarrow & & \downarrow \\ Y & \longrightarrow & \bigcirc Y \end{array}$$

is a pullback since f is $\bigcirc$-étale. Since the exponential functor preserves limits, the left and right faces of the cube are pullbacks. By Lemma 1.28 the back face is a pullback, and i and f are orthogonal. $\square$

Chapter 3

Nullification

We now turn our attention to an important class of modalities which lend themselves to geometric sitautions. These modalities are known as **nullification** at a collection of types.

Definition 3.1. Let $D : A \to \mathcal{U}$ be a type family. We say X is D**-null** if the map

$$x \mapsto (d \mapsto x) : X \to (D(a) \to X)$$

is an equivalence for each $a : A$. That is every map $D(a) \to X$ is constant.

Nullification is an operation which takes a type X and produces a D-null type functorially. The D-null types are intuitively those that "see" each type in the collection D as contractible. Nullification appears in many places: We will see it can be used to generalise the propositional truncation operation introduced in Chapter 1. It has also been used to incorparate concepts from synthetic differential geometry into HoTT [Mye22b].

3.1 Constructing Nullification

To construct nullification we will use a higher inductive type to force all the conditions needed to hold.

Definition 3.2. Let $D : A \to \mathcal{U}$ and $X : \mathcal{U}$ we define the type $N_D(X)$ to be the HIT with constructors:

1. $|-| : X \to N_D(X)$
2. $\texttt{hub} : \{a : A\} \to (D(a) \to N_D(X)) \to N_D(x)$
3. $\texttt{spoke} : \{a : A\}\ (f : D(a) \to N_D(X))\ (d : D(a)) \to \texttt{hub}(f) = f(d)$

4. $\texttt{hub}_= : \{x, y : N_D(X)\}\ \{a : A\} \to (D(a) \to x = y) \to x = y$

5. $\texttt{spoke}_= : \{x, y : N_D(X)\}\ \{a : A\}\ (f : D(a) \to x = y)\ (d : D(a)) \to \texttt{hub}_=(f) = f(d)$

We will show the first constructor satisfies the correct properties of a modal unit. The second and third constructors tell us that the image of every map $f : D(a) \to N_D(X)$ is contractible. The map `hub` gives us the centre of contraction, and `spoke` gives us the contraction onto `hub`. Similarly the fourth and fifth condition tells us that for every pair of elements $x, y : N_D(X)$ any map $f : D(a) \to x = y$ has contractible image.

We will now show that we have a Σ-closed reflective subuniverse given by nullification. To do this we will need an alternate description of equivalences.

Definition 3.3. A map $f : A \to B$ is a **path-split equivalence** if:

- f has a right inverse.
- For all $x, y : A$ the map $\texttt{ap}_f : x = y \to f(x) = f(y)$ has a right inverse.

Lemma 3.4. *A map $f : A \to B$ is an equivalence iff it is a path-split equivalence.*

Proof. Suppose f is an equivalence. Then f has an inverse, say g, so f clearly has a right inverse. Also $\texttt{ap}_f$ also has an inverse given by $\texttt{ap}_g$, which is also a right inverse. So f is a path-split equivalence.

Now suppose f is a path-split equivalence. Then let $g : B \to A$ be the right inverse. We claim g is also a left inverse. Let $x : A$. Then $f(g(f(x))) = f(x)$ since g is a right inverse to f. By applying the right inverse of $\texttt{ap}_f$, we deduce $g(f(x)) = x$. Thus f is an equivalence. □

We will require the following simple lemma about function extensionality:

Lemma 3.5. *Let $x, y : X$ and $q : x = y$. We have a term $\texttt{funExt}(\lambda d.q) : \lambda d.x = \lambda d.y$. This satisfies $\texttt{funExt}(\lambda d.q) = \texttt{ap}_{\lambda x.\lambda d.x}(q)$.*

Proof. By path induction it is enough to show this result when q is refl_x. In this case we have $\texttt{funExt}^-(\text{refl}_{\lambda d.x}) :\equiv \lambda x.\, \text{refl}_x$. So $\texttt{funExt}(\lambda x.\, \text{refl}_x) = \text{refl}_{\lambda d.x} \equiv \texttt{ap}_{\lambda x.\lambda d.x}(\text{refl}_x)$. □

Lemma 3.6. *For all $D : A \to \mathcal{U}$ and $X : \mathcal{U}$ the type $N_D(X)$ is D-null.*

Proof. We need to show that for each $a : A$ the map

$$x \mapsto (d \mapsto x) : N_D(X) \to (D(a) \to N_D(X))$$

is an equivalence. We will show that it is a path-split equivalence. So we start by constructing a right inverse to this map. Define

$$\phi : (D(a) \to N_D(X)) \to N_D(X)$$

$$\phi(f) :\equiv \mathtt{hub}(f)$$

This is a right inverse since

$$(\lambda d.\phi(f) = f) \simeq \left(\prod_{d:D(a)} \mathtt{hub}(f) = f(d) \right)$$

by function extensionality, and $\mathtt{spoke}(f)$ has the type of the right hand side.

Now let $x, y : N_D(X)$. We will show that the induced map

$$\mathtt{ap}_{\lambda x.\lambda d.x} : x = y \to (\lambda d.x) = (\lambda d.y)$$

has a right inverse. Given $p : \lambda d.x = \lambda d.y$ we obtain

$$\mathtt{funExt}^-(p) : \prod_{d':D(a)} ((\lambda d.x)(d') = (\lambda d.y)(d'))$$

The type reduces judgementally to give:

$$\mathtt{funExt}^-(p) : D(a) \to (x = y)$$

Then we define the right inverse ψ to be $\psi(p) :\equiv \mathtt{hub}_=(\mathtt{funExt}^-(\mathtt{p}))$. This is a right inverse since for any $p : \lambda d.x = \lambda d.y$ we have $\mathtt{funExt}^-(p) : D(a) \to x = y$ so we can apply $\mathtt{spoke}_=$ to get:

$$\mathtt{spoke}_=(\mathtt{funExt}^-(p)) : \prod_{d:D} \mathtt{hub}_=(\mathtt{funExt}^-(p)) = \mathtt{funExt}^-(p, d)$$

Noting the value on the left is definitionally the application of a constant function, using function extensionality we obtain:

$$\mathtt{funExt}(\mathtt{spoke}_=(\mathtt{funExt}^-(p))) : \lambda d.\mathtt{hub}_=(\mathtt{funExt}^-(p)) = \mathtt{funExt}^-(p)$$

Finally we apply $\texttt{funExt}$, and using Lemma 3.5, we get:

$$\begin{aligned}\texttt{ap}_{\lambda x.\lambda d.x}(\psi(p)) &\equiv \texttt{ap}_{\lambda x.\lambda d.x}(\texttt{hub}_{=}(\texttt{funExt}^{-}(p))) \\ &= \texttt{funExt}(\lambda d.\texttt{hub}_{=}(\texttt{funExt}^{-}(p))) \\ &= \texttt{funExt}(\texttt{funExt}^{-}(p)) \\ &= p\end{aligned}$$

□

Lemma 3.7. *Nullification forms a reflective subuniverse*

Proof. We choose the modal operator to be N_D and the unit to be $|-|$. The modal family are the D-null types. We have already shown that $N_D(X)$ is D-null. Thus it only remains to show the map $f \mapsto f \circ |-| : (N_D(X) \to Y) \to (X \to Y)$ is an equivalence for every D-null type Y. This follows by using the induction principle on $N_D(X)$ to extend a map $X \to Y$ to a map $N_D(X) \to Y$ when Y is D-null. To see this proof see Spitters, Shulman, and Rijke, Theorem 2.18 [SSR20]. □

Lemma 3.8. *Nullification is a Σ-closed reflective subuniverse.*

Proof. Suppose $X : \mathcal{U}$ and $Y : X \to \mathcal{U}$ satisfy X is D-null and $Y(x)$ is D-null for all $x : X$. Then for all $a : A$ we have

$$\begin{aligned}D(a) \to \sum_{x:X} Y(x) &\simeq \sum_{g:D(a)\to X} \prod_{d:D(a)} Y(g(d)) && \text{Separating into components} \\ &\simeq \sum_{x:X} D(a) \to Y(x) && \text{Since } X \text{ is } D\text{-null} \\ &\simeq \sum_{x:X} Y(x) && \text{Since } Y(x) \text{ is } D\text{-null}\end{aligned}$$

So $\sum_{x:X} Y(x)$ is D-null. □

For notational ease we refer to properties of the modality N_D by just the type D. For example we will say a map $f : X \to Y$ is D-étale rather than N_D-étale.

3.2 Properties

Lemma 3.9. *Let $D : A \to \mathcal{U}$ be a type family. For each $a : A$ the type $N_D(D(a))$ is contractible.*

Proof. We choose the centre of contraction to be $\mathtt{hub}(|-|)$, where

$$|-| : D(a) \to N_D(D(a))$$

is the constructor. We want to show $\mathtt{hub}(|-|) = z$ for all $z : N_D(D(a))$. Since $N_D(D(a))$ is N_D-modal, so is the type $\mathtt{hub}(|-|) = z$ for any $z : N_D(D(a))$. Thus it suffices to show $\mathtt{hub}(|-|) = |d|$ for each $d : D(a)$, by modal induction. We have $\mathtt{spoke}(|-|, d) : \mathtt{hub}(|-|) = |d|$ for each $d : D(a)$. Then by definition $N_D(D(a))$ is contractible. □

Now fix a pointed type (D, d_0) and consider the modality N_D. This is notation for nullification at the constant type family $\mathbf{1} \to \mathcal{U}$ given by $* \mapsto D$.

Lemma 3.10. *All propositions are D-null.*

Proof. Let P be a proposition. Then $\lambda p.\lambda d.p : P \to (D \to P)$ and $\lambda f.f(d_0) : (D \to P) \to P$ are inverse. This is because P and $D \to P$ are both propositions, so there is nothing to check. We verify $D \to P$ is a proposition. If $f, g : D \to P$ then for all $x : D$ we have $f(x) = g(x)$ since P is a proposition. By function extensionality $f = g$. □

Lemma 3.11. *All the modal units $\eta_X : X \to N_D(X)$ are surjective*

Proof. We wish to define a function $(z : N_D(X)) \to \|(y : X) \times (|y| = z)\|$. Since the codomain is a proposition, by Lemma 3.10 it is D-null. Hence it is enough to define a function $x : X \to \|(y : X) \times (|y| = |x|)\|$ by modal induction. The map $x \mapsto |(x, \mathrm{refl}_{|x|})|$ works. □

Lemma 3.12. *For any pointed type (X, x) we have*

$$(D, d_0) \to_* (X, x) \simeq (D, d_0) \to \mathbb{D}^D(X, x)$$

Proof. The backwards map is simply projection onto the first argument. That is for $(g, p) : (D, d_0) \to_* \mathbb{D}^D(X, x)$, we get $\mathtt{pr}_1 \circ g : B \to X$. And since $p : g(d_0) = (x, \mathrm{refl}_{|x|})$ we deduce that $\mathtt{ap}_{\mathtt{pr}_1}(p) : \mathtt{pr}_1(g(d_0)) = x$

The forwards map is defined in the following way. Suppose we have $(f, p) : (D, d_0) \to_* (X, x)$ Note that $N_D(D)$ is contractible, by Lemma 3.9. Hence for all $d, d' : D$ we have $|d| = |d'|$. Then by Lemma 2.13 we deduce that $|f(d)| = |f(d')|$. But then for all $d : D$ we have $|f(d)| = |f(d_0)| = |x|$. And so we can extend f to $\tilde{f} : D \to \mathbb{D}^D(X, x)$. And since the proof $|f(d_0)| = |f(d_0)|$ is given by reflexivity, this becomes a pointed map.

Checking these two operations are inverse to each other is an exercise in definition unwinding. □

We now provide some additional motivation for nullification. Nullification is an extremely important class of modalities, generalising some of the most common modalities. For example propositional truncation is nullification at S^0. We will show that nullifying at S^{n+1} generalises this construction, and provides the operation of n-truncation, removing the higher homotopical information from a type.

Lemma 3.13 ([Uni13] Theorem 7.2.9)**.** *A type A is n-truncated iff for all $a : A$ we have $\Omega^{n+1}(A, a)$ is contractible.*

Proposition 3.14. *A type is n-truncated iff it is S^{n+1}-null.*

Proof. A type A is n-truncated iff for all $a : A$ the type $\Omega^{n+1}(A, a)$ is contractible, by Lemma 3.13. But by Corollary 1.21 we have $\Omega^{n+1}(A, a) \simeq S^{n+1} \to_* (A, a)$. This and Lemma 3.13 imply if A is n-truncated then $S^{n+1} \to A \simeq (a : A) \times (S^{n+1} \to_* (A, a)) \simeq A$. This equivalence is given by evaluation at the basepoint so A is S^{n+1}-null.

Now suppose A is S^{n+1}-null. Then $(S^{n+1} \to A) \simeq A$ by evaulation at the basepoint. But then the projection map $(a : A) \times (S^{n+1} \to_* (A, a)) \to A$ is an equivalence. Hence the fibres of this map, given by $S^{n+1} \to_* (A, a)$, must be contractible. So for each a, the loop space $\Omega^{n+1}(A, a)$ is contractible. □

We refer to S^{n+1}-nullification as **n-truncation** and denote it $\|\_\|_n$, extending propositional truncation.

3.3 Étale maps for nullification

A simple corollary of the factorisation system of Theorem 2.27 is that all D-étale maps lift against the inclusion of the distinguished point of D. This is because it is clear that $\mathbf{1} \to D$ is a D-equivalence since $N_D(D)$ is contractible.

Quesiton 3.15. *Are the D-étale maps precisely those which are right orthogonal to the map $\mathbf{1} \to D; * \mapsto d_0$?*

The current status of this question is open, although some partial results have been answered. We now develop the theory to answer this question in the special case of the modality given by nullifying at S^{n+1} the $n+1$-sphere [CR20]. We will present this special case in a much more general way than originally described with the hope that this method of proof can be extended. We incorparate ideas of Christensen et al. who define pairs of modalities with a similar relationship to nullification at the n-sphere and $(n+1)$-sphere. [Chr+18].

Definition 3.16. Let $(\texttt{isModal}, \bigcirc, \eta)$ be a modality.

1. We say a map $f : X \to Y$ is **$\bigcirc$-modal** if for all $y : Y$ we have $\texttt{isModal}(\mathrm{fib}_f(y))$.
2. We say a type X is **$\bigcirc$-separated** if for all $x, y : X$ the type $x = y$ is modal.
3. We say a type X is **$\bigcirc$-connected** if $\bigcirc X$ is contractible. Further a map $f : X \to Y$ is **$\bigcirc$-connected** if for all $y : Y$ the fiber $\mathrm{fib}_f(y)$ is $\bigcirc$-connected.

It turns out for any modality, the collection of $\bigcirc$-separated types forms a modality, however we will only show this in the case of nullification. Now fix a type B, which is not necessarily pointed. Nullification at B and at its suspension ΣB hold a special relation to each other: The ΣB-null types are precisely the B-separated types.

Lemma 3.17. *A type X is ΣB-null iff it is B-separated*

Proof. X is B-separated iff for all $x, y : X$ we have $x = y \simeq (B \to x = y)$ by the $p \mapsto \lambda b.p$. But this is true iff

$$X \simeq \sum_{x,y:X} (x = y) \simeq \sum_{x,y:X} (B \to x = y)$$

by the map $x \mapsto (x, x, \lambda b.\, \mathrm{refl}_x)$. The recursion principle for suspensions states

$$\sum_{x,y:X} (B \to x = y) \simeq \Sigma B \to X$$

so we deduce X is be B-seperated iff $X \simeq \Sigma B \to X$ by the map $x \mapsto \lambda z.x$. That is X is ΣB-null. □

Lemma 3.18. *We have for all types X an equivalence $N_D(N_{\Sigma D}(A)) \simeq N_D(A)$*

Proof. This is an immediate corollary of Christensen et al., Lemma 2.11 [Chr+18]. □

Lemma 3.19. *Fix a type X. For all $x, y : X$ we have $N_B(x = y) \simeq |x| =_{\Sigma B} |y|$.*

Proof. We adapt the proof of Theorem 7.3.12 from "Homotopy Type Theory" [Uni13]. Call the subuniverse of B-null types $\mathcal{U}_B$. We have seen that equality types between modal types are modal. Thus by definition the type $\mathcal{U}_B$ is B-separated and is hence ΣB-null. Now define a type family by modal induction:

$$P : N_{\Sigma B}(X) \to N_{\Sigma B}(X) \to \mathcal{U}_B$$

$$P(|x|, |y|) :\equiv N_B(x = y)$$

Then for all $u, v : N_{\Sigma B}(X)$ we define a map $\mathtt{decode} : P(u, v) \to u = v$ by induction again. We define this map on $|x|, |y|$ and $p : x = y$ to be

$$\mathtt{decode}(|p|) :\equiv \mathtt{ap}_{|-|}(p)$$

For all $x : X$ we have $P(|x|, |x|) \equiv N_B(x = x)$ and thus $|\operatorname{refl}_x| : P(|x|, |x|)$ Since for $u : N_{\Sigma B}(X)$ the type $P(u, u)$ is B-null, it is thus ΣB-null and we may apply modal induction to define $\phi : \prod_{u:N_{\Sigma B}} P(u, u)$. Then we may define an inverse map for $u, v : N_{\Sigma B}(X)$. By transporting along the type family $\lambda z.P(u, z)$ and $p : u = v$ we obtain a function $p_* : P(u, u) \to P(u, v)$. Then we define:

$$\mathtt{encode} : u = v \to P(u, v)$$

$$\mathtt{encode}(p) = p_*(\phi(u)) : P(u, v)$$

The maps $\mathtt{decode}$ and $\mathtt{encode}$ are inverse. To show $\mathtt{decode}(\mathtt{encode}(p)) = p$ for all $p : u = v$ it is enough to check when p is refl_u by path induction. The type $u = u$ is ΣB-null, so by modal induction we can assume $u \equiv |x|$ for some $x : X$. But then

$$\begin{aligned}
\mathtt{decode}(\mathtt{encode}(\operatorname{refl}_{|x|})) &\equiv \mathtt{decode}(\operatorname{refl}_{|x|_*}(\phi(|x|)) \\
&\equiv \mathtt{decode}(|\operatorname{refl}_x|) && \text{since } \operatorname{refl}_{|x|_*} \text{ is the identity} \\
&\equiv \mathtt{ap}_{|-|}(\operatorname{refl}_x) \\
&\equiv \operatorname{refl}_{|x|}
\end{aligned}$$

and we deduce that $\mathtt{decode} \circ \mathtt{encode}$ is the identity.

For all $u, v : N_{\Sigma B}(X)$ we want to show $\prod_{z:P(u,v)} \mathtt{encode}(\mathtt{decode}(z)) = z$. Since $P(u, v)$ is B-null and equalities preserve being modal, the type which forms our goal is B-null and hence is ΣB-null. So we may assume $u = |x|$ and $v = |y|$ for some $x, y : X$ by induction. Applying induction again we may assume $z = |p| : N_B(x = y) \equiv P(|x|, |y|)$, for some $p : x = y$. Now we may apply path induction on p to assume $x \equiv y$ and $p \equiv \operatorname{refl}_x$. Then we may compute, purely judgementally:

$$\begin{aligned}
\mathtt{encode}(\mathtt{decode}(|\operatorname{refl}_x|)) &\equiv \mathtt{encode}(\mathtt{ap}_{|-|}(\operatorname{refl}_x)) \\
&\equiv \mathtt{encode}(\operatorname{refl}_{|x|}) \\
&\equiv \operatorname{refl}_{|x|_*}(\phi(|x|)) \\
&\equiv \phi(|x|) \\
&\equiv |\operatorname{refl}_x|
\end{aligned}$$

Thus we have shown for all $u, v : N_{\Sigma B}(X)$ the types $P(u, v)$ and $u = v$ are equivalent. So we deduce for all $x, y : X$ the types $N_B(x = y) :\equiv P(|x|, |y|) \simeq |x| = |y|$. □

Lemma 3.20. *Suppose X, Y are ΣB-connected. Then any $f : X \to Y$ is B-connected.*

Proof. Let $y : Y$ Then

$$\begin{aligned} N_B(\mathrm{fib}_f(y)) &\equiv N_B\left(\sum_{x:X} f(x) = y\right) \\ &\simeq N_B\left(\sum_{x:X} N_B(f(x) = y)\right) && \text{By Lemma 2.10} \\ &\simeq N_B\left(\sum_{x:X} \eta_{\Sigma B}(f(x)) = \eta_{\Sigma B}(y)\right) && \text{By Lemma 3.19} \\ &\simeq N_B(X) \\ &\simeq \mathbf{1} \end{aligned}$$

The penultimate equivalence is because Y is ΣB-connected, and thus the type $\eta_{\Sigma B}(f(x)) = \eta_{\Sigma B}(y)$ is contractible. The final equivalence is because X is ΣB-connected and hence is B-connected. Hence by definition f is B-connected. □

Definition 3.21. We say a type B is **good** if for any map $f : X \to Y$ between connected types such that $f^{\Sigma B} : X^{\Sigma B} \to Y^{\Sigma B}$ is an equivalence, we have f is N_B-modal.

As part of the proof of Cherubini and Rijke, Theorem 3.10 [CR20] it is remarked that S^n is **good**. In fact a stronger result is claimed, that the converse is true, that whenever f is S^n-modal, the map $f^{S^{n+1}}$ is an equivalence. This strengthened claim is false, however, and we provide a counterexample.

Example 3.22. Consider the unique map $S^1 \to \mathbf{1}$. This is S^2-modal, since the circle is 1-truncated. Now we can exponentiate this by $\Sigma S^2 \equiv S^3$ to get a map $(S^1)^{S^3} \to (\mathbf{1})^{S^3}$. Now $(\mathbf{1})^{S^3} \simeq \mathbf{1}$. And S^1 is 1-truncated and thus is 2-truncated. Hence S^1 is S^3-null and hence $(S^1)^{S^3} \simeq S^1$. So this gives a map $S^1 \to \mathbf{1}$, which clearly cannot be an equivalence.

Theorem 3.23. *Suppose B is good. Then a map $f : X \to Y$ is ΣB-étale iff it lifts against the base point inclusion onto the basepoint $N : \Sigma B$.*

Proof. The forwards direction follows from Theorem 2.27, since the map $\mathbf{1} \to \Sigma B$ is a ΣB-equivalence.

For the reverse direction, suppose the map $f : X \to Y$ lifts against the basepoint inclusion. That is the square

$$\begin{array}{ccc} X^{\Sigma B} & \xrightarrow{f^{\Sigma B}} & Y^{\Sigma B} \\ \downarrow & & \downarrow \\ X & \xrightarrow{f} & Y \end{array}$$

is a pullback. The downwards maps are evaluation at the basepoint N. So the fibre at $x : X$ of $X^{\Sigma B} \to X$ is

$$\left(\sum_{f:\Sigma B \to X} f(N) = x \right) \equiv (\Sigma B, N) \to_* (X, x)$$

Similarly for $y : Y$ the fibre of $Y^{\Sigma B} \to Y$ is $(\Sigma B, N) \to_* (Y, y)$ Since this is a pullback the induced map on fibres are equivalences by Lemma 1.27. Hence for each $x : X$ the induced map

$$((\Sigma B, N) \to_* (X, x)) \to ((\Sigma B, N) \to_* (Y, f(x)))$$

is an equivalence. Now by Lemma 3.12 applied to the source and target, we obtain a map

$$((\Sigma B, N) \to_* D^{\Sigma B}(X, x)) \to ((\Sigma B, N) \to_* D^{\Sigma B}(Y, f(x)))$$

which is an equivalence. But we may check that this map is induced by

$$df_x : D^{\Sigma B}(X, x) \to D^{\Sigma B}(Y, f(x))$$

Since B is good we deduce the map df_x is B-modal. But also the disks $D^{\Sigma B}(X, x)$ and $D^{\Sigma B}(Y, f(x))$ are ΣB-connected. Thus by Lemma 3.20 the map df_x is B-connected. A B-connected and B-modal map is an equivalence. Thus for all $x : X$ the map df_x is an equivalence. Using Lemma 1.27 we deduce that the following square is a pullback:

$$\begin{array}{ccc} T^{\Sigma B}X & \xrightarrow{T^{\Sigma B}f} & T^{\Sigma B}Y \\ \downarrow & & \downarrow \\ X & \xrightarrow{f} & Y \end{array}$$

Since ΣB is pointed, the modal units are surjective by Lemma 3.11, we may apply Lemma 2.18 and deduce that f is ΣB-étale. □

Corollary 3.24. *A map $f : X \to Y$ is $\| - \|_n$-étale iff it lifts against the basepoint inclusion $\mathbf{1} \to S^{n+1}$*

Proof. Since S^n for any $n \geq 0$ is good we may apply Theorem 3.23 and deduce the result. □

References

[BDR18] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. *Higher Groups in Homotopy Type Theory*. 2018. arXiv: 1802.04315 [cs.LO].

[BR21] Ulrik Buchholtz and Egbert Rijke. *The long exact sequence of homotopy n-groups*. 2021. arXiv: 1912.08696 [math.AT].

[Che18] Felix Cherubini. *Cartan Geometry in Modal Homotopy Type Theory*. 2018. DOI: 10.48550/ARXIV.1806.05966. URL: https://arxiv.org/abs/1806.05966.

[Chr+18] J. Daniel Christensen et al. "Localization in Homotopy Type Theory". In: (2018). DOI: 10.48550/ARXIV.1807.04155. URL: https://arxiv.org/abs/1807.04155.

[CR20] Felix Cherubini and Egbert Rijke. "Modal descent". In: *Mathematical Structures in Computer Science* 31.4 (2020), pp. 363–391. DOI: 10.1017/s0960129520000201. URL: https://doi.org/10.1017%5C%2Fs0960129520000201.

[Gra+21] Daniel Gratzer et al. "Multimodal Dependent Type Theory". In: *Logical Methods in Computer Science* Volume 17, Issue 3 (July 2021). DOI: 10.46298/lmcs-17(3:11)2021. URL: https://doi.org/10.46298%5C%2Flmcs-17%5C%283%5C%3A11%5C%292021.

[HS98] Martin Hofmann and Thomas Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998, pp. 83–111.

[KL18] Chris Kapulkin and Peter LeFanu Lumsdaine. *The Simplicial Model of Univalent Foundations (after Voevodsky)*. 2018. arXiv: 1211.2851 [math.LO].

[KS17] Igor Khavkine and Urs Schreiber. *Synthetic geometry of differential equations: I. Jets and comonad structure*. 2017. DOI: 10.48550/ARXIV.1701.06238. URL: https://arxiv.org/abs/1701.06238.

[Mye22a] David Jaz Myers. *Good Fibrations through the Modal Prism*. 2022. arXiv: 1908.08034 [math.CT].

[Mye22b] David Jaz Myers. *Orbifolds as microlinear types in synthetic differential cohesive homotopy type theory*. 2022. DOI: 10.48550/ARXIV.2205.15887. URL: https://arxiv.org/abs/2205.15887.

[nLa] nLab authors. *local diffeomorphism*. https://ncatlab.org/nlab/show/local+diffeomorphism. Date Accessed: Apr. 2023.

[Pen81] Jacques Penon. "Infinitésimaux et intuitionnisme". fre. In: *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 22.1 (1981), pp. 67–72. URL: http://eudml.org/doc/91254.

[Rij19] Egbert Rijke. *Classifying Types*. 2019. arXiv: 1906.09435 [math.LO].

[Sch13] Urs Schreiber. *Differential cohomology in a cohesive infinity-topos*. 2013. arXiv: 1310.7930 [math-ph].

[Shu15] Michael Shulman. *Brouwer's fixed-point theorem in real-cohesive homotopy type theory*. 2015. DOI: 10.48550/ARXIV.1509.07584. URL: https://arxiv.org/abs/1509.07584.

[Shu19] Michael Shulman. *All* $(\infty, 1)$*-toposes have strict univalent universes*. 2019. arXiv: 1904.07004 [math.AT].

[SSR20] Bas Spitters, Michael Shulman, and Egbert Rijke. "Modalities in homotopy type theory". In: *Logical Methods in Computer Science* 16 (2020).

[Sta23] The Stacks project authors. *The Stacks project*. https://stacks.math.columbia.edu. 2023.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[Voe06] Vladimir Voevodsky. "A very short note on homotopy λ-calculus". In: *Unpulbished* (Sept. 2006), 1–7. URL: https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf.