Synthetic Homotopy Theory and Classifying Principal Bundles in Homotopy Type Theory

Candidate number:

1004569

University of Oxford

A dissertation submitted towards the degree of

Master of Mathematics and Computer Science

Trinity 2019

Contents

Introduction

This dissertation is intended to provide an introduction to homotopy type theory (HoTT) with a view towards doing homotopy theory in this framework. We will illustrate its use in this context by providing a novel proof in HoTT of a result with applications in the theory of principal bundles due originally to Daniel Gottlieb [Got69]. Our version of the theorem takes the form:

Theorem. *Let G be a discrete group, X be a 0-connected type, and $f : X \to K(G, 1)$ be a pointed map. Then there is an equivalence*

$$Map(X, K(G, 1); f) \simeq K(C_G(im\pi_1 f), 1)$$

where $Map(X, K(G, 1); f)$ is the connected component of the mapping space $X \to K(G, 1)$ containing f, $C_G(S)$ denotes the centraliser of $S \subseteq G$, and $K(H, n)$ denotes an Eilenberg-MacLane space.

Building towards understanding the statement and proof of this theorem is the main purpose of the dissertation.

Homotopy type theory is a new foundations of mathematics based on Martin-Löf intensional type theory [ML75]. Our main reference is the Homotopy Type Theory Book [Uni13], which was the product of numerous authors following the Institute for Advanced Studies' Special Year on the Univalent Foundations of Mathematics in 2012-13. One of the main ideas, originally explored by Steve Awodey and Michael Warren in [AW09], is that intensional type theory can be equipped with a topological interpretation: types A become spaces and terms $a : A$ become points in those spaces. Witnesses $p : (a =_A b)$ of equality between two points $a : A$ and $b : A$ become paths from a to b in A. Moreover, witnesses $s : (p =_{(a=_A b)} q)$ to equality between paths become homotopies or 2-paths, and iterating this idea yields 3-paths, 4-paths, and so on. These identifications help to resolve the difficulties presented by the complex structure of equality in intensional type theory. Additional insights which make HoTT

into a viable place to do mathematics are Voevodsky's *univalence axiom* [Voe06], and the ability to define *higher inductive types* which turns many objects of study into native concepts in HoTT.

As a framework, homotopy type theory inherits desirable properties from intensional type theory. In particular, proofs in HoTT are amenable to verification using proof assistants such as Coq [INR] or AGDA [Nor]. Moreover, it encourages use of type-theoretic techniques which may lead to a proof taking on a very different flavour compared to a more classical argument, potentially yielding additional insight into a problem. One constraint is that, by its nature, all constructions in HoTT must be homotopy invariant. While this means that we must occasionally alter definitions away from their classical counterparts, it often also means that proofs are not dependent on concrete realisations, and can be applied in a number of different contexts.

Content Outline In Chapter 1 we provide a brief introduction to Martin-Löf intensional type theory, including many of the basic constructions that we will need. We then show how the *identity types* $(a =_A b)$ of witnesses to equality endow types with the structure of a *weak* ∞*-groupoid*, facilitating the homotopy interpretation. The rest of the chapter focusses on presenting the definitions and results that we will need in the rest of the dissertation, including the *univalence axiom*.

Chapter 2 follows up by providing various constructions and results from homotopy theory in HoTT. In particular, these constitute a *synthetic approach* to homotopy theory. To illustrate doing homotopy theory in this way, we also spend some time examining the proof that $\pi_1(\mathbb{S}^1) = \mathbb{Z}$.

Finally, in Chapter 3 we seek to apply the theory from the preceding chapters to understand the definition of principal bundles in homotopy type theory and classify them. We give a presentation of fibre bundles and principal bundles in HoTT, before proving a series of lemmas which will allow us to prove the main theorem as stated at the start of this introduction.

Chapter 1

Homotopy Type Theory

We begin by giving a brief introduction to homotopy type theory, describing the basic structure of the underlying intensional type theory, before introducing the homotopy interpretation of identity types and Voevodsky's univalence axiom. This is intended to give a broad overview of the theory to readers unfamiliar with (homotopy) type theory: for a more in-depth review, the Homotopy Type Theory book [Uni13] is the definitive reference.

1.1 Martin-Löf Intensional Type Theory

Rather than present Martin-Löf type theory [ML75] as a formal type system, we will take a more intuitive approach. A **type** can be considered as a collection of **terms** or **inhabitants**. We write $a : A$ to mean that a belongs to A, or that the term a has type A. However, we note a key distinction between type theory and set theory. In type theory, the statement $a : A$ is a **judgement**: being of type A is an inherent property of the term a, and this is derivable in the type system. While the similar statement $a \in A$ in set theory is a **proposition**: it has a truth value. The difference is made clear when working internally. We can write things like "if $a \in A$, then $b \in B$" as expressions within set theory, but in type theory there is no expression corresponding to "if $a : A$ then $b : B$". Nor can we try to prove or disprove $a : A$. A typing judgement is just something that holds in theory or doesn't.

This distinction between judgements and propositions crops up again when we look at equality between terms. In particular this is an *intensional* type theory where we have both **judgemental equality**, written $a \equiv b$, and **propositional equality**, "$a = b$". Judgemental equality says that two terms are equal by definition, and indeed we will use that notation when defining certain terms. If $a : A$ and $a \equiv b$ hold

in the type theory, then it is immediate that $b : A$. If a, b both belong to the same type A, then the question of propositional equality between a and b manifests itself as a type $a =_A b$. If this type has an inhabitant, then a and b are propositionally equal, otherwise they are not. Proving that $a : A$ and $b : A$ are propositionally equal is therefore equivalent to providing an element of $a =_A b$. Note that if terms $a : A$ and $b : B$ don't belong to the same type then they can't be equal (in either sense). From now on we will say "equal" to mean propositionally equal, and "equal by definition" (or something similarly qualified) to mean judgementally equal.

This idea of a type A representing a proposition and an inhabitant $a : A$ representing a proof, or witness to the truth of that proposition, can be applied more generally. The constructions we introduce in this section will allow us to perform logical operations within this **propositions-as-types** interpretation, which we shall mention as we introduce them.

Types may belong to other types. In particular, we have a **universe** of types $\mathcal{U}$ whose inhabitants are "all types". For the theory to be consistent, $\mathcal{U}$ can't contain itself, so behind the scenes there is a cumulative hierarchy of universes $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : ...$, and for our purposes we let $\mathcal{U}$ be the $\mathcal{U}_i$ which contains all the types we need.

We will now consider several type constructors.

1.1.1 Function Types

Non-Dependent Functions Given types A and B, we can form the type $A \to B$ of (non-dependent) **functions**, or **maps**, from A to B. In set theory a function is a certain kind of relation. However, in type theory they are a primitive concept. We can introduce a function by giving it a name, f say, and then defining $f : A \to B$ with an equation

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

where Φ is some expression possibly containing the variable x. Alternatively we can avoid having to give a function a name by using λ notation

$$(\lambda(x : A).\Phi).$$

In either case, we compute the value of the function at some $a : A$ via substitution

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

where $\Phi[a/x]$ is the expression obtained by replacing all free occurrences of x in Φ with a. That is, the computation rule is just β-reduction. Clearly such a function is only well defined if $\Phi[a/x] : B$ for all $a : A$. As in λ-calculus, $\to$ binds to the right: so the type of functions of several variables can be written in the form $A \to B \to C$. The type theory also has η-reduction, so that $f \equiv \lambda x.f(x)$. We may also use other common notations for functions like $x \mapsto \Phi$. The logical interpretation of $A \to B$ in the propositions-as-types interpretation is "A implies B".

Dependent Functions Martin-Löf type theory also allows us to define functions whose codomain can contain inhabitants of multiple different types. Given a type $A : \mathcal{U}$ and a **type family** $B : A \to \mathcal{U}$, we can construct the type of **dependent functions** $\prod_{(x:A)} B(x)$. A dependent function is introduced in much the same way as a non-dependent function: $f : \prod_{x:A} B(x)$ is defined via an equation $f(x) :\equiv \Phi$ and evaluating at $a : A$ is defined via substitution. The only difference is that for f to be well-defined, we now require that $\Phi[a/x] : B(a)$ for each $a : A$. Note that if B is a constant function $x \mapsto C$, then $\prod_{(x:A)} B(x) \equiv (A \to C)$. The logical interpretation of $\prod_{(x:A)} B(x)$ in the propositions-as-types interpretation is "for all $x : A$, $B(x)$ holds".

1.1.2 Pair Types

Cartesian Product (Non-Dependent Pairs) Given types A and B, we form their **Cartesian product** type $A \times B$. Inhabitants of this type are of the form (a, b) constructed from inhabitants $a : A$ and $b : B$. The non-dependent elimination rule (also known as the **recursion principle**) says that for any function $g : A \to B \to C$, we can define a function $f : A \times B \to C$ such that

$$f((a, b)) :\equiv g(a)(b).$$

In particular, we define the two projection functions

$$pr_1((a, b)) :\equiv a$$

$$pr_2((a, b)) :\equiv b.$$

The dependent elimination rule (the **induction principle**) says to define a dependent function $f : \prod_{x:A\times B} C(x)$ we can provide a map $g : \prod_{(x:A)} \prod_{(y:B)} C((x, y))$, and then $f((a, b)) :\equiv g(a)(b)$. The logical interpretation of $A \times B$ in the propositions-as-types interpretation is "A and B".

Dependent Pairs As in the case of functions, we generalise pairs to allow the type of the second component to vary. Given a type $A : \mathcal{U}$ and type family $B : A \to \mathcal{U}$, we form the type of **dependent pairs** $\sum_{(x:A)} B(x)$. Inhabitants are constructed by pairing $a : A$ with $b : B(a)$. The recursion principle takes a dependent function $g : \prod_{(x:A)}(B(x) \to C)$ and provides a function $f : (\sum_{(x:A)} B(x)) \to C$ defined

$$f((a, b)) :\equiv g(a)(b).$$

The induction principle takes a dependent function $g : \prod_{(x:A)} \prod_{(y:B(x))} C((x, y))$ and produces a dependent function $f : \prod_{p:\sum_{(x:A)} B(x)} C(p)$ with the same defining equation. In this case we need to use the induction principle to define pr_2, since when $(a, b) : \sum_{(x:A)} B(x)$, the map $(a, b) \mapsto b$ is dependent. The logical interpretation of $\sum_{(x:A)} B(x)$ in the propositions-as-types interpretation is "there exists $x : A$ such that $B(x)$ holds". In the absence of bracketing, both $\prod$ and $\sum$ scope over the rest of a type expression.

1.1.3 Coproduct Types

Given types A and B, we form the **coproduct** type $A + B$ which can be thought as the disjoint union of A and B. There are constructors $inl : A \to A + B$ ("left injection") and $inr : B \to A + B$ ("right injection"). The recursion principle takes functions $g_l : A \to C$ and $g_r : B \to C$, and provides a function $f : A + B \to C$ defined by cases

$$f(inl(a)) :\equiv g_l(a)$$

$$f(inr(b)) :\equiv g_r(b).$$

The induction principle takes dependent functions $g_l : \prod_{(x:A)} C(inl(x))$ and $g_r : \prod_{(y:B)} C(inr(y))$, and provides a dependent function $f : \prod_{(x:A+B)} C(x)$ defined by the same equations. The logical interpretation of $A + B$ in the propositions-as-types interpretation is "A or B".

1.1.4 Empty and Unit Types

Empty Type The **empty type 0** is the type with no inhabitants. It has no constructors, and always has a unique non-dependent or dependent function from it to any other type or type family with no defining equations. The logical interpretation of $\mathbf{0}$ in the propositions-as-types interpretation is "false". We interpret the function type $A \to \mathbf{0}$ as "not A".

Unit Type The **unit type 1** is the type with a single inhabitant which is given by the constant constructor $\star : \mathbf{1}$. Non-dependent functions $f : \mathbf{1} \to C$ and dependent functions $f : \prod_{(x:\mathbf{1})} C(x)$ are constructed by providing an inhabitant $c : C$ or $c : C(\star)$. Then $f(\star) :\equiv c$. The logical interpretation of **1** in the propositions-as-types interpretation is "true".

1.1.5 Natural Numbers

As a concrete example, we introduce the type of **natural numbers** $\mathbb{N}$. The constructors are

$$0 : \mathbb{N}$$

$$succ : \mathbb{N} \to \mathbb{N}.$$

The recursion and induction principles are now what we might expect from their names. The recursion principle takes an inhabitant $c_0 : C$ and a function $g : \mathbb{N} \to C \to C$ and provides a function $f : \mathbb{N} \to C$ defined by

$$f(0) :\equiv c_0$$

$$f(succ(n)) :\equiv g(n, f(n)).$$

Given a type family $C : \mathbb{N} \to \mathcal{U}$, the induction principle takes $c_0 : C(0)$ and a dependent function $g : \prod_{(n:\mathbb{N})} C(n) \to C(succ(n))$ and provides a dependent function $f : \prod_{(n:\mathbb{N})} C(n)$ defined by the same equations.

From the natural numbers we can construct other useful types, such as the **integers** $\mathbb{Z}$ or the **real numbers** $\mathbb{R}$. We will not provide these constructions, but will later make use of the integers with functions *succ* and *pred*.

1.1.6 Identity Types

We mentioned at the start of this section that the question of whether or not two inhabitants $a, b : A$ of a type A are equal implicates a type $a =_A b$. This is an **identity type** and we have just described its formation rule. The introduction rule is reflexivity: we know that any two things which are definitionally the same are equal, yielding a constructor $refl : \prod_{(x:A)} x =_A x$. In particular, if $a \equiv b$, then $refl_a : a =_A b$, since $a =_A b$ is also judgementally equivalent to the type $a =_A a$. The induction rule for identity types is important for the homotopy interpretation. In fact, we will give it a name

Definition 1.1.1 (Path Induction). Suppose we have a family

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

and a function

$$c : \prod_{x:A} C(x, x, refl_x),$$

then there is a function

$$f : \prod_{x,y:A} \prod_{p:x=_A y} C(x, y, p)$$

such that

$$f(x, x, refl_x) \equiv c(x).$$

In the propositions-as-types interpretation it says that to prove a property for all terms $x : A$, $y : A$ and witnesses to their equality $p : x =_A y$, it suffices to prove that property when $x \equiv y$ and the witness is $refl_x$. We also note that there is an equivalent version called **based path induction** which is the same except that x is fixed.

1.2 The Homotopy Interpretation

To motivate the homotopy interpretation, we first see how types take on the structure of an **∞-groupoid**. Recall that a **groupoid** is a category in which every morphism is an isomorphism. Using path induction, we can show that if we take the terms of a type A as "objects" and inhabitants of $x =_A y$ as the "morphisms" from object x to object y, then A has the structure of a groupoid. In particular, given $x, y, z : A$, we use path induction to construct functions

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

$$p \mapsto q \mapsto p \cdot q$$

and

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

$$p \mapsto p^{-1}$$

such that $refl_X \cdot refl_x \equiv refl_x$ and $(refl_x)^{-1} \equiv refl_x$, called **composition** and the **inverse** respectively. We still need some coherence conditions: for example, for A to look like a category, we need composition to be associative. Given $p : w =_A x$, $q : x =_A y$ and $r : y =_A z$, it's not true in general that $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$.

However, using path induction again we show that the identity type $p \cdot (q \cdot r) =_{(w =_A z)} (p \cdot q) \cdot r$ is inhabited. Think of this like a "2-morphism" between the two morphisms. The other conditions we want (like $p \cdot p^{-1} = refl_w$) can also be proved using path induction, so A really does look like a groupoid. In fact, it is a **weak** groupoid since the conditions like the associativity of composition are only given up to "higher morphisms". But this weakness gives us additional structure: concatenation and inverses of 2-morphisms obey coherence conditions given by 3-morphisms, operations on these obey coherence conditions given by 4-morphisms, which obey coherence conditions given by 5-morphisms and so on. In this way we see that identity types have the structure of weak ∞-groupoids.

Another context where ∞-groupoids turn up is in the **fundamental ∞-groupoid** $\Pi_\infty X$ of a topological space X. This has the points of X as objects, paths between points as the morphisms, homotopies between paths as the 2-morphisms, homotopies between homotopies as the 3-morphisms and so on. Moreover, the *homotopy hypothesis* implicates an equivalence of categories **Top** $\xrightarrow{\Pi_\infty}$ ∞-**Grp** via a (bi)adjunction between Π_∞ and the geometric realisation construction. The idea, therefore, is to interpret types with their ∞-groupoid structure like topological spaces. Terms $x : A$ become points, witnesses $p : x =_A y$ become paths from x to y, witnesses $s : p =_{(x =_A y)} q$ become homotopies between paths (or 2-paths), and so on. Under this interpretation, composition can be seen as concatenation of paths and inversion is path reversal.

1.3 Equivalence, Functoriality and Transport

There are several equivalent notions of when two types are equivalent. We shall focus on the definition via **quasi-inverse** maps. But first we need the following definition.

Definition 1.3.1. Let f, g be dependent maps of type $\prod_{x:A} P(x)$ for some type family $P : A \to \mathcal{U}$. A **homotopy** between f and g is a dependent map of type

$$(f \sim g) :\equiv \prod_{x:A} (f(x) = g(x)).$$

Definition 1.3.2. Suppose $f : A \to B$ is a map of types. A **quasi-inverse** to f is a triple (g, α, β) consisting of a map $g : B \to A$ and homotopies $\alpha : f \circ g \sim id_B$ and $\beta : g \circ f \sim id_A$.

If $f : A \to B$ has a quasi-inverse, we say that f is an **equivalence**, the types A and B are **equivalent**, and write $A \simeq B$.

Lemma 1.3.1 (2-out-of-3 rule) ([Uni13] Theorem 4.7.1). *Suppose $f : A \to B$ and $g : B \to C$. If any two of f, g and $g \circ f$ are equivalences, then so is the third.*

We now look at how functions act functorially on paths. In particular we have

Lemma 1.3.2 ([Uni13] Lemma 2.2.1). *Suppose that $f : A \to B$ is a function. Then for any $x, y : A$ there is a function*

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

such that for each $x : A$, we have $ap_f(refl_x) \equiv refl_{f(x)}$.

Proof. By path induction, defining $ap_f(refl_x) :\equiv refl_{f(x)}$. □

Moreover, ap behaves functorially.

Lemma 1.3.3 ([Uni13] Lemma 2.2.2). *Given functions $f : A \to B$, $g : B \to C$ and paths $p : x =_A y$, $q : y =_A z$, we have*

1. $ap_f(p \cdot q) = ap_f(p) \cdot ap_f(q)$
2. $ap_f(p^{-1}) = ap_f(p)^{-1}$
3. $ap_g(ap_f(p)) = ap_{g \circ f}(p)$
4. $ap_{id_A}(p) = p$

The function ap_f ensures that if $(x =_A y)$ is inhabited, then so is $(f(x) =_B f(y))$. More generally, given any type family $B : A \to \mathcal{U}$ and dependent function $f : \prod_{a:A} B(a)$, we should have that $f(x)$ and $f(y)$ are also equal in some sense. However, these objects belong to types $B(x)$ and $B(y)$ respectively, which might not be the same, so it doesn't make sense to talk about the identity type "$f(x) = f(y)$". The solution is to make use of a witness path $p : (x =_A y)$ and lift this to a path in $\sum_{a:A} B(a)$. We will call this **transport** and have the following lemmas.

Lemma 1.3.4 ([Uni13] Lemma 2.3.1) - Transport. *Suppose that $P : A \to \mathcal{U}$ is a type family, and that $p : x =_A y$. Then there is a function $p_* : P(x) \to P(y)$. We will also write p_* as $transport^P(p, -)$.*

Proof. By path induction, defining $(refl_x)_* :\equiv id_{P(x)} : P(x) \to P(x)$. □

Lemma 1.3.5 ([Uni13] Lemma 2.3.2) - Path Lifting Property. *Suppose* $P : A \to \mathcal{U}$ *is a type family, and* $u : P(x)$ *for some* $x : A$*. Then for any* $p : x =_A y$*, we have a path*

$$lift(u, p) : (x, u) = (y, p_*(u))$$

in $\sum_{a:A} P(a)$ *such that* $pr_1(lift(u, p)) = p$*.*

Proof. By based path induction, with $C : \prod_{y:A}(x =_A y) \to \mathcal{U}$ defined $C :\equiv \lambda y \lambda p.((x, u) = (y, p_*(u)))$ and by noticing that $C(x, refl_x) \equiv ((x, u) = (x, (refl_x)_*(u))) \equiv ((x, u) = (x, u))$, which is inhabited by $refl_{(x,u)}$. □

At this point it is helpful to recall that a **fibration**, in the context of topology, is a continuous map $E \to B$ from some total space into a base space which satisfies the homotopy lifting property. Lemma 1.3.5 gives us a way of lifting paths for all type families $P : A \to \mathcal{U}$, so in fact, we can think of P as a fibration over base space A with total space $\sum_{a:A} P(a)$ and fibres given by $P(a)$ for $a : A$.

We can now give the following version of Lemma 1.3.2 for dependent functions.

Lemma 1.3.6 ([Uni13] Lemma 2.3.4). *Suppose* $f : \prod_{a:A} P(a)$*. Then there is a map*

$$apd_f : \prod_{p:x=y} (p_*(f(x)) =_{P(y)} f(y))$$

We will also introduce the notation $(u =_p^P v) :\equiv (transport^P(p, u) = v)$*, so that the above can be written* $apd_f : \prod_{p:x=y}(f(x) =_p^P f(y))$*.*

There are some useful additional lemmas about *transport* which show that it respects both path and function composition.

Lemma 1.3.7 ([Uni13] Lemma 2.3.9). *Given* $P : A \to \mathcal{U}$ *with* $p : x =_A y$*,* $q : y =_A z$*, and* $u : P(x)$*, we have*

$$transport^P(q, transport^P(p, u)) = transport^P(p \cdot q, u)$$

Lemma 1.3.8 ([Uni13] Lemma 2.3.10). *Given* $f : A \to B$ *and* $P : B \to \mathcal{U}$ *along with some* $p : x =_A y$ *and* $u : P(f(x))$*, we have*

$$transport^{P \circ f}(p, u) = transport^P(ap_f(p), u)$$

1.4 Univalence

In this section we introduce Voevodsky's **univalence axiom**, which is part of what makes the homotopy interpretation so powerful. It's a form of extensionality principle for universe types: "types that behave in the same way (i.e. are equivalent) are equal". Note first that if two types are equal in $\mathcal{U}$, then they are equivalent.

Lemma 1.4.1 ([Uni13] Lemma 2.10.1). *Given types $A, B : \mathcal{U}$, we have a function*

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

given by $p \mapsto transport^{X \mapsto X}(p, -) \equiv p_*$, where $X \mapsto X$ denotes the type family $id_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}$.

Univalence then takes on the following form.

Definition 1.4.1 (Univalence Axiom). Given types $A, B : \mathcal{U}$, the map $idtoeqv : (A =_{\mathcal{U}} B) \to (A \simeq B)$ is an equivalence with inverse

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

so that $transport^{X \mapsto X}(ua(f), x) = f(x)$, and $ua(transport^{X \mapsto X}(p, -)) = p$.

An important consequence of univalence is function extensionality.

Theorem 1.4.2 ([Uni13] Theorems 4.9.4 / 4.9.5) - Function Extensionality. *Given two dependent maps f, g of type $\prod_{x:A} P(x)$ for some type family $P : A \to \mathcal{U}$, there is an equivalence*

$$(f = g) \simeq (f \sim g)$$

given by quasi-inverse maps $happly : (f = g) \to (f \sim g)$ and $funext : (f \sim g) \to (f = g)$.

1.5 Homotopy Truncation

As a final preliminary, we introduce the notion of truncation. The idea here is that given a type A and an integer $n \geq -2$, we can kill off all the homotopy structure of A above level n. Firstly however, we provide a few definitions of what it means to have no homotopy structure above a given level.

Definition 1.5.1. A type A is **contractible** if there is some $a : A$ such that $a = x$ is inhabited for all $x : A$. In particular we have an inhabitant of

$$isContr(A) \equiv \sum_{a:A} \prod_{x:A} (a = x).$$

In this case a is called the **centre of contraction**.

An immediate consequence of being contractible is the following.

Lemma 1.5.1 ([Uni13] Lemma 3.11.3). *A type A is contractible iff $A \simeq \mathbf{1}$.*

Definition 1.5.2. We define a map $\textit{is-n-type} : \mathcal{U} \to \mathcal{U}$ for $n \geq -2$ by the following recursion:

$$\textit{is-n-type}(A) :\equiv \begin{cases} isContr(A) & \text{if } n = -2, \\ \prod_{x,y:A} \textit{is-m-type}(x =_A y) & \text{if } n = m + 1 \end{cases}$$

If $\textit{is-n-type}(A)$ is inhabited we say that A is an n**-type** or n**-truncated**. In particular, for $n = 0$, we say that A is a **set**, or **discrete**.

Lemma 1.5.2 ([Uni13] Theorem 7.1.9). *Let $n \geq -2$, $A : \mathcal{U}$ and $B : A \to \mathcal{U}$. If $B(a)$ is an n-type for each $a : A$, then so is $\prod_{x:A} B(x)$.*

Lemma 1.5.3 ([Uni13] Theorem 7.1.11). *Let $n\textit{-type} :\equiv \sum_{A:\mathcal{U}} \textit{is-n-type}(A)$ denote the type of all n-types. Then n-type is an $(n+1)$-type.*

We now introduce the n-truncation of a type A. We won't provide the explicit construction since we don't need it, but will briefly talk about its recursion principle.

Definition 1.5.3. Suppose A is a type, and $n \geq -2$. Then there is a type $\|A\|_n$ called the n**-truncation** of A, whose inhabitants are generated from those of A via a map $|-|_n : A \to \|A\|_n$.

Lemma 1.5.4 ([Uni13] Lemma 7.3.1). *$\|A\|_n$ is an n-type.*

The recursion principle for truncations is particularly useful, and therefore has a name.

Lemma 1.5.5 ([Uni13] Lemma 7.3.3) - Universal Property of Truncations. *Let $n \geq -2$, $A : \mathcal{U}$, and B be an n-type, then we have an equivalence*

$$(\|A\|_n \to B) \simeq (A \to B)$$

given by the map $g \mapsto g \circ |-|_n$.

This says that when we are considering maps into an n-type, we can ignore any homotopy structure of the domain above the n-th level.

The (-1)-truncation is also referred to as **propositional truncation** since $\|A\|_{-1}$ has a single inhabitant iff A is inhabited, and is otherwise empty. The 0-truncation of a type gives its distinct path-connected components. The truncation also allows us to define the extent to which a type is **connected**:

Definition 1.5.4. Let $n \geq -2$. A type A is **n-connected** if $\|A\|_n$ is contractible.

Being n-connected is dual in a sense to being an n-type. n-types have no homotopy structure above level n; n-connected types have no homotopy structure at or below level n.

Chapter 2

Synthetic Homotopy Theory

Having introduced the basic ideas of homotopy type theory, we're finally in a position to start talking about what it's good for. We will focus on its application to homotopy theory. In particular, since objects from homotopy theory like points, spaces (types), paths and homotopies are "native" concepts in homotopy type theory, we can apply a *synthetic approach* rather than an *analytic approach*. As an analogy, compare the synthetic, axiomatic geometry of Euclid to the analytic point-set geometry of Descartes.

As mentioned in the introduction, there are a number of arguments for why doing homotopy theory in this synthetic way is a good idea. One of the best is that this approach suggests new type-theoretic methods for proving theorems. While some proofs are essentially direct transcriptions of classical proofs, many are much more computer-science flavoured, using induction-like reasoning. We will see this throughout the chapter, where we examine some of the different constructions and results of synthetic homotopy theory.

2.1 Homotopy Groups

We begin with one of the most fundamental constructions, which takes its structure directly from that of the identity types of a space.

Definition 2.1.1. Given a pointed type (X, x_0), its **loop space**, denoted $\Omega(X, x_0)$ is the pointed type $(x_0 =_X x_0, refl_{x_0})$ of loops in X based at x_0.

We may define the $n^{\mathbf{th}}$**-iterated loop spaces** inductively.

$$\Omega^0(X, x_0) :\equiv (X, x_0)$$
$$\Omega^{n+1}(X, x_0) :\equiv \Omega^n(\Omega(X, x_0))$$

Definition 2.1.2. Given a pointed type (X, x_0), its $n^{\mathbf{th}}$ **homotopy group** is

$$\pi_n(X, x_0) :\equiv \|\Omega^n(X, x_0)\|_0$$

where the group operation is induced by path composition. For $n = 0$ we actually define $\pi_0(X) :\equiv \|X\|_0$ without reference to a basepoint, noting that in this case we do not generally get any group structure.

In this work we will only really consider discrete groups: i.e a 0-type equipped with a group operation. These behave exactly as they do in set-theoretic mathematics. As we'd expect, the n-th homotopy groups for $n \geq 2$ are all Abelian, which can be shown via the Eckmann-Hilton argument.

Theorem 2.1.1 ([Uni13] Theorem 2.1.6) - Eckmann-Hilton. *For $n \geq 2$, the composition operator on the n-th loop space is commutative.*

The homotopy groups of homotopy type theory enjoy many of the same algebraic tools as the classically-defined groups. They are functorial: given a pointed map $f : X \to Y$, there is a path $f_0 : f(x_0) = y_0$. So we can construct $\Omega f : \Omega X \to \Omega Y$ as $p \mapsto f_0^{-1} \cdot ap_f(p) \cdot f_0$. Truncating this yields a map $\pi_1 f : \pi_1 X \to \pi_1 Y$, and we can iterate to get maps $\pi_k f$ for all $k \geq 0$ ($\pi_0 f$ is just the truncation of f). We also have long exact sequences.

Definition 2.1.3. Given a map $f : X \to Y$ and $y : Y$, the **fibre** of f at y is the type

$$fib_f(y) :\equiv \sum_{x:X} (f(x) = y).$$

The **image** of f is the type

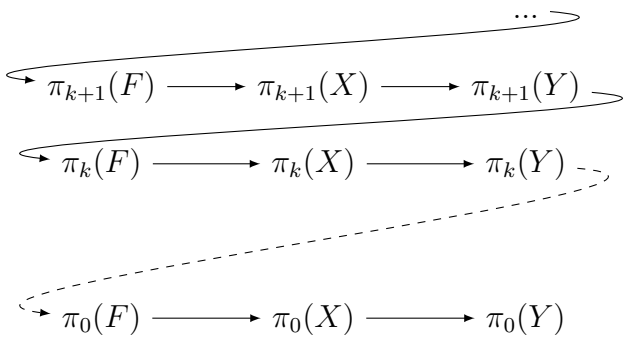
$$im(f) :\equiv \sum_{y:Y} \sum_{x:X} (f(x) = y) \equiv \sum_{y:Y} fib_f(y).$$

If Y is a pointed type with basepoint y_0, then the **kernel** of f is the type

$$ker(f) :\equiv \sum_{x:X} (f(x) = y_0) \equiv fib_f(y_0).$$

When X and Y are sets, so too are $im(f)$ and $ker(f)$. In this case we say that a sequence of maps (f_i) is **exact** precisely when $ker(f_i) = im(f_{i-1})$ as sets for each i.

Theorem 2.1.2 ([Uni13] Theorem 8.4.6) - Long exact sequence of a fibration. *Let $f : X \to Y$ be a basepoint-preserving map between pointed spaces with fibre $F :\equiv fib_f(y_0)$. Then we have the following long exact sequence*

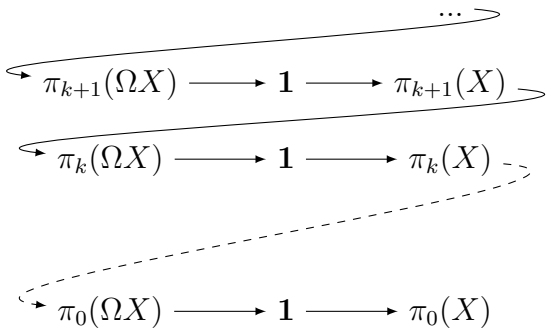


The maps on each rows are of the form $\pi_k f$ or $\pi_k i$ where i is the inclusion of F into X. The connecting map $\pi_1 Y \to \pi_0 F$ is $|r|_0 \mapsto |(x_0, f_0 \cdot r)|_0$.

One use for this is following lemma relating the homotopy groups of a type X and its loop space ΩX.

Lemma 2.1.3. *For any pointed type (X, x_0), we have $\pi_k(\Omega(X, x_0)) \simeq \pi_{k+1}(X, x_0)$*

Proof. We note that the fibre over x_0 of the map $pr_1 : (\sum_{x:X} x_0 = x) \to X$ is $\Omega(X, x_0)$. But now the type $\sum_{x:X} x_0 = x$ is contractible since, given any $(x, p) : \sum_{x:X} x_0 = x$, we have that $p : x_0 = x$ and $p_*(refl_{x_0}) = p$, so that $(x, p) = (x_0, refl_{x_0})$. Hence, using Lemma 2.1.2, we obtain a long exact sequence of the form:



from which we immediately see that $\pi_k(\Omega X) \simeq \pi_{k+1}(X)$. □

Loop spaces also play nicely with truncations:

Lemma 2.1.4 ([Uni13] Corollary 7.3.13). *Let $n \geq -2$ and (X, x_0) be a pointed type. Then*

$$\|\Omega(X, x_0)\|_n = \Omega(\|(X, x_0)\|_{n+1})$$

2.2 The Circle: $\mathbb{S}^1$

In this section we will define the circle type $\mathbb{S}^1$ as a **higher inductive type** (HIT) and examine its homotopy structure. Higher inductive types turn up in a number of places later on and essentially give a schema for defining types. In the same way that we introduced the basic type formers of Martin-Löf type theory in Section 1.1, a HIT is defined by constructors and elimination rules. One difference, and the reason for the appearance of the word "higher", is that the constructors are not constrained to generate points of the type. They may also generate paths between points and higher paths.

Definition 2.2.1. The circle $\mathbb{S}^1$ is defined as the higher inductive type generated by

- A point $base : \mathbb{S}^1$
- A path $loop : base =_{\mathbb{S}^1} base$

as illustrated in Figure 2.2, with induction principle:

> Given $P : \mathbb{S}^1 \to \mathcal{U}$, $b : P(base)$ and $\ell : b =^P_{loop} b$, there is some $f : \prod_{x:\mathbb{S}^1} P(x)$ with $f(base) \equiv b$ and $apd_f(loop) = \ell$.

from which we can derive the recursion principle:

> Given B with $b : B$ and $\ell : b = b$, we have $f : \mathbb{S}^1 \to B$ with $f(base) \equiv b$ and $ap_f(loop) = \ell$.

Homotopy groups Following the exposition in [Uni13] Section 8.1, we will prove that the fundamental group $\pi_1(\mathbb{S}^1) = \mathbb{Z}$, and that the higher homotopy groups $\pi_n(\mathbb{S}^1)$ are trivial for $n \geq 2$. Recalling the definition

$$\pi_n(A, a) :\equiv \|\Omega^n(A, a)\|_0$$

we see that we will obtain both results by showing that $\Omega(\mathbb{S}^1) = \mathbb{Z}$, since then $\Omega(\mathbb{S}^1)$ is a set, so not only are all higher loop spaces are trivial, but also $\|\mathbb{Z}\|_0 = \mathbb{Z}$. We start by imitating the classical proof, but then present a more type-theoretic *encode-decode* style proof.

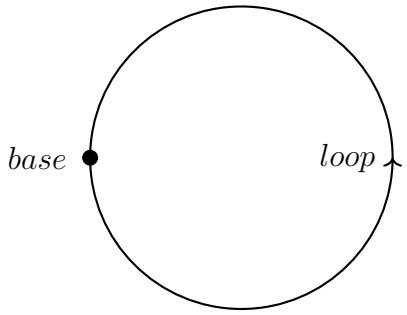


Figure 2.1: $\mathbb{S}^1$ as a higher inductive type generated by a single point and a path from that point to itself.

Classical proof A 'classical' homotopy-theoretic proof might go as follows:

> Consider the *winding map* $w : \mathbb{R} \to \mathbb{S}^1$ given by $w(t) \coloneqq e^{2\pi i t}$. We can show that this is a fibration of $\mathbb{R}$ over $\mathbb{S}^1$. In fact, this is the universal cover of the circle. Similarly, if $P_{base}\mathbb{S}^1$ is the space of based paths in $\mathbb{S}^1$, then the map $e : P_{base}\mathbb{S}^1 \to \mathbb{S}^1$ which sends a path p to its endpoint is a fibration of $P_{base}\mathbb{S}^1$ over $\mathbb{S}^1$. We observe that $\mathbb{R} \simeq P_{base}\mathbb{S}^1$ since both are contractible (we can retract any path along itself to the constant path at *base*). But now we make use of the fact that a map of fibrations is a homotopy equivalence iff it's a fibre-homotopy equivalence. Hence we see that, in particular, the fibres of the two fibrations over *base* are homotopy equivalent: $\mathbb{Z} = w^{-1}(base) \simeq e^{-1}(base) = \Omega(\mathbb{S}^1)$.

To see how this proof might be translated into our framework, recall that in Section 1.3 we saw how we could interpret a type family $P : B \to \mathcal{U}$ as a fibration. Translating the path fibration $e : P_{base}\mathbb{S}^1 \to \mathbb{S}^1$ in the above proof gives the family $\lambda(x : \mathbb{S}^1).(base =_{\mathbb{S}^1} x)$. Translating the winding map is a little more involved. The key thing to observe is that marching anticlockwise round the circle once is the same as adding 1 in the stalk. In this vein, we note that $succ : \mathbb{Z} \to \mathbb{Z}$ is an equivalence. Hence, by univalence, there is a path $ua(succ) : \mathbb{Z} =_{\mathcal{U}} \mathbb{Z}$ in $\mathcal{U}$. By the recursion principle for $\mathbb{S}^1$ there is a function $code : \mathbb{S}^1 \to \mathcal{U}$ such that $code(base) \equiv \mathbb{Z}$, and $ap_{code}(loop) = ua(succ)$. Now we see that $\sum_{x:\mathbb{S}^1} code(x)$ behaves like the universal cover, in that the fibre over *base* is $\mathbb{Z}$, and traversing the circle yields the successor

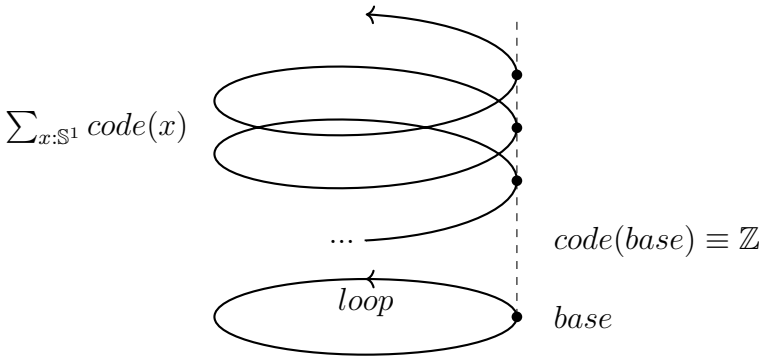


Figure 2.2: Universal cover of $\mathbb{S}^1$. Transporting a trip round *loop* into the cover takes you to the successor in the fibre.

in the fibre.

$$\begin{aligned}
transport^{code}(loop, x) &= transport^{A \mapsto A}(ap_{code}(loop), x) && \text{(by Lemma 1.3.8)} \\
&= transport^{A \mapsto A}(ua(succ), x) && \text{(by definition of } code\text{)} \\
&= idtoeqv(ua(succ))(x) && \text{(by definition of } idtoeqv\text{)} \\
&= succ(x) && \text{(by univalence)} \\
&= x + 1
\end{aligned}$$

Similarly, we have that $transport^{code}(loop^{-1}, -) = pred$. So *code* is the appropriate translation of the universal cover. See Figure 2.2 for an illustration.

Equipped with our type-theoretic versions of the universal cover and path fibration, we define a map $f : \prod_{x:\mathbb{S}^1}(base =_{\mathbb{S}^1} x) \to code(x)$ as

$$f_x(p) :\equiv transport^{code}(p, 0)$$

To complete the proof we'd want to show that this function induces an equivalence on the total spaces, $\sum_{x:\mathbb{S}^1}(base =_{\mathbb{S}^1} x)$ and $\sum_{x:\mathbb{S}^1} code(x)$, since these are both contractible, and that therefore f is an equivalence on each fibre. In particular, that $f(base, -)$ is an equivalence $(base =_{\mathbb{S}^1} base) \simeq code(base)$ $(\equiv \mathbb{Z})$.

While there are ways to carry out the proofs indicated above, but instead we will look at the more type-theoretic encode-decode argument.

Encode-decode proof This proof, due to Daniel Licata and Mike Shulman [LS13], again makes use of the function $f : \prod_{x:\mathbb{S}^1}(base =_{\mathbb{S}^1} x) \to code(x)$ we defined in the

previous section, but now we will call it *encode*. Recall therefore the definition

$$encode_x(p) :\equiv transport^{code}(p, 0)$$

We see that since *transport* is functorial, if p is of the form $(loop \cdot loop^{-1} \cdot loop \cdot ...)$, then $transport^{code}(p, -)$ will act as a composition of the form $(succ \circ pred \circ succ \circ ...)$. So in fact $encode_{base}$ computes the winding number of a path from *base* to *base*.

The difference with this proof compared to the classical one is that, instead of trying to show the types $\sum_{x:\mathbb{S}^1}(base =_{\mathbb{S}^1} x)$ and $\sum_{x:\mathbb{S}^1} code(x)$ are contractible, we will instead set up an explicit quasi-inverse to *encode*. With that in mind, let's define a function $loop^- : \mathbb{Z} \to (base =_{\mathbb{S}^1} base)$ as follows

$$loop^n = \begin{cases} \overbrace{loop \cdot loop \cdot ... \cdot loop}^{n} & \text{if } n > 0, \\ \underbrace{loop^{-1} \cdot loop^{-1} \cdot ... \cdot loop^{-1}}_{-n} & \text{if } n < 0, \\ refl_{base} & \text{if } n = 0. \end{cases}$$

We now define our inverse map $decode : \prod_{x:\mathbb{S}^1} code(x) \to (base =_{\mathbb{S}^1} x)$ by the induction principle for $\mathbb{S}^1$. Recall that the induction principle for $\mathbb{S}^1$ with $P(x) :\equiv code(x) \to (base =_{\mathbb{S}^1} x)$ requires $b : P(base)$ and $\ell : b =^P_{loop} b$ $(\equiv transport^P(loop, b))$. We let $b :\equiv loop^-$ and give ℓ as the path defined by the following sequence of equations (see [Uni13] Section 8.1.4 for details):

$$\begin{aligned} &transport^{x \mapsto code(x) \to (base = x)}(loop, loop^-) \\ &\qquad = transport^{x \mapsto (base = x)}(loop, -) \circ loop^- \circ transport^{code}(loop^{-1} -) \\ &\qquad = (- \cdot loop) \circ loop^- \circ transport^{code}(loop^{-1}, -) \\ &\qquad = (- \cdot loop) \circ loop^- \circ pred \\ &\qquad = (n \mapsto loop^{n-1} \cdot loop) \\ &\qquad = (n \mapsto loop^n) \\ &\qquad \equiv loop^- \end{aligned}$$

Lemma 2.2.1. *Given $x : \mathbb{S}^1$ and $p : base =_{\mathbb{S}^1} x$, we have $decode_x(encode_x(p)) = p$.*

Proof. By path induction it suffices to show that $decode_{base}(encode_{base}(refl_{base})) = refl_{base}$. But $encode_{base}(refl_{base}) \equiv transport^{code}(refl_{base}, 0) \equiv 0$ and $decode_{base}(0) \equiv loop^0 \equiv refl_{base}$. □

Lemma 2.2.2. *Given $x : \mathbb{S}^1$ and $c : code(x)$, we have that $encode_x(decode_x(c)) = c$.*

Proof. By circle induction this reduces to showing that $encode_{base}(loop^n) = n$ for each $n : \mathbb{N}$, which we do by induction over $\mathbb{N}$. In the base case, we saw in the proof of Lemma 2.2.1 that $encode_{base}(loop^0) \equiv encode_{base}(refl_{base}) \equiv 0$. For the induction step we see that $encode_{base}(loop^{n+1}) = encode_{base}(loop^n \cdot loop) = transport^{code}(loop^n \cdot loop, 0) = transport^{code}(loop, transport^{code}(loop^n, 0)) = succ(transport^{code}(loop^n, 0))$ which equals $succ(n)$ by the induction hypothesis. □

Now we are done, since Lemmas 2.2.1 and 2.2.2 show that $encode_x$ and $decode_x$ are quasi-inverse for each $x : \mathbb{S}^1$. In particular, the case for the fibres over $x \equiv base$ yields an equivalence $\Omega(\mathbb{S}^1) \simeq \mathbb{Z}$, as desired.

2.3 Suspensions

Having looked at the circle, we would like to define higher-dimensional spheres. However, rather than defining $\mathbb{S}^n$ for each n individually, we construct them inductively from $\mathbb{S}^1$. The construction we need is the suspension ΣA of a space A. In the classical setting, we usually define ΣA by taking a product of A with the unit interval I and shrinking the copies of A at the endpoints down to points. But the ability to specify path constructors for HITs suggests the following definition.

Definition 2.3.1. Given a type A, the **suspension** ΣA of A is the higher inductive type generated by

- A point $N : \Sigma A$
- A point $S : \Sigma A$
- A function $merid : A \to (N =_{\Sigma A} S)$

with induction principle:

> Given $P : \Sigma A \to \mathcal{U}$, $n : P(N)$, $s : P(S)$ and $m : A \to (n =^P_{merid} s)$, there is a function $f : \prod_{x:\Sigma A} P(x)$ such that $f(N) = n$, $f(S) = s$ and $apd_f(merid(a)) = m(a)$ for all $a : A$.

and recursion principle:

> Given a type B with $n, s : B$ and $m : A \to (n =_B s)$, there is a function $f : \Sigma A \to B$ such that $f(N) = n$, $f(S) = s$ and $ap_f(merid(a)) = m(a)$ for all $a : A$.

The names and types of the constructors should suggest defining the 2-sphere as $\Sigma\mathbb{S}^1$, with north pole N, south pole S, and an $\mathbb{S}^1$-indexed family of meridian curves between the poles as in Figure 2.3. In fact, we define all n-spheres in this way.

Definition 2.3.2. For $n \geq 2$ the **n-sphere** $\mathbb{S}^n$ is $\Sigma^n\mathbb{S}^1$.

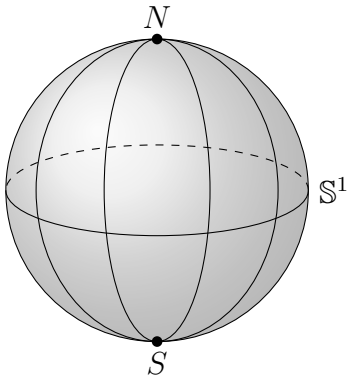


Figure 2.3: Diagram showing $\mathbb{S}^2$ as the suspension $\Sigma\mathbb{S}^1$ with an $\mathbb{S}^1$-indexed family of meridian paths.

Using the recursion principle, we see that Σ acts as a functor. In the context of pointed types, it is left adjoint to Ω.

Definition 2.3.3. Given types X and Y, we will use the notation $Map(X, Y)$ to refer to the type $X \to Y$. If X and Y are pointed, by x_0 and y_0 say, we define the type of **pointed maps** between X and Y as

$$Map_*(X, Y) :\equiv \sum_{f:Map(X,Y)} f(x_0) = y_0.$$

Lemma 2.3.1 ([Uni13] Lemma 6.5.4) - Suspension-loop space adjunction. *Given pointed types A and B we have*

$$Map_*(\Sigma A, B) \simeq Map_*(A, \Omega B)$$

2.4 Wedge Sums and Smash Products

The $\Sigma \dashv \Omega$ adjunction is in fact a special case of a more general adjunction which expresses a form of currying. But we will need a few definitions, beginning with the following definition of pushouts in homotopy type theory.

Definition 2.4.1. Given maps $f : A \to B$ and $g : A \to C$, the **pushout** of f and g is the higher inductive type $B \sqcup^A C$ generated by

- A function $inl : B \to B \sqcup^A C$
- A function $inr : C \to B \sqcup^A C$
- A function $glue : A \to (inl(f(a)) = inr(g(a))$

expressing the commutative diagram:

$$\begin{array}{ccc} A & \xrightarrow{g} & C \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle inr} \\ B & \xrightarrow{inl} & B \sqcup^A C \end{array}$$

with recursion principle:

> Given type D, maps $l : B \to D$, $r : C \to D$, and $g : \prod_{a:A}(l(f(a)) = r(g(a))$, there is a map $s : B \sqcup^A C \to D$ such that $s \circ inl = l$, $s \circ inr = r$ and $ap_s \circ glue = g$.

As an example, we see that the suspension ΣA is the pushout of $\mathbf{1} \leftarrow A \to \mathbf{1}$. We also have the following types expressed as pushouts.

Definition 2.4.2. Given two pointed types (A, a_0) and (B, b_0), the **wedge sum** $A \vee B$ is the pushout of

$$A \xleftarrow{a_0} \mathbf{1} \xrightarrow{b_0} B.$$

Define the inclusion map $i^{\vee}_{A,B} : A \vee B \to A \times B$ using the recursion principle for pushouts with $l(a) :\equiv (a, b_0)$, $r(b) :\equiv (a_0, b)$, and $g(\star) :\equiv refl_{(a_0,b_0)}$. Then the **smash product** $A \wedge B$ is the pushout of

$$\mathbf{1} \leftarrow A \vee B \xrightarrow{i^{\vee}_{A,B}} A \times B.$$

The wedge sum can be viewed as the space obtained by gluing together the two spaces at their basepoint. As an example, Figure 2.4 shows a wedge sum of the 2-sphere and the circle.

The smash product can be thought of as the usual Cartesian product of two spaces where a copy of their wedge sum has been shrunk to a point. It has the structure of a symmetric monoidal product (see Guillaume Brunerie's thesis [Bru16] for details). Moreover, much like the symmetric monoidal tensor product in the context of rings, it exposes a currying adjunction as expressed in the following result (see Andreas Franz's masters thesis [Fra17] for a proof).

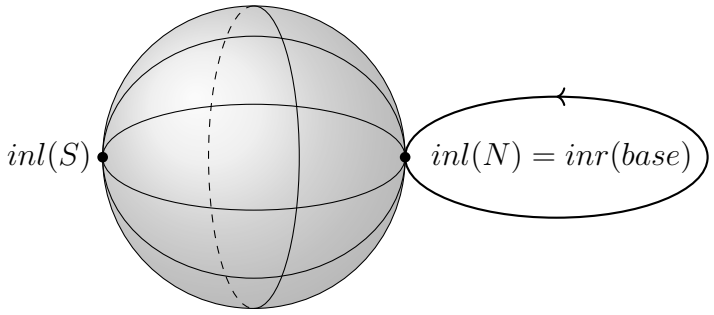


Figure 2.4: Illustration of the wedge sum $(\mathbb{S}^2, N) \vee (\mathbb{S}^1, base)$.

Lemma 2.4.1 ([Fra17] Proposition 6.7). *If $(X, x_0), (Y, y_0), (Z, z_0)$ are pointed types, then we have an equivalence*

$$Map_*(X, Map_*(Y, Z)) \simeq Map_*(X \wedge Y, Z)$$

We defined higher-dimensional spheres as suspensions of $\mathbb{S}^1$, but had we defined spheres independently, it turns out that we can form suspensions by taking the smash product with them. In particular, we have the following result given by Brunerie.

Lemma 2.4.2 ([Bru16] Proposition 4.2.1). *Given any pointed type (X, x_0), we have an equivalence*

$$\mathbb{S}^1 \wedge X \simeq \Sigma X$$

Corollary 2.4.2.1. *Given any pointed type (X, x_0) and $n : \mathbb{N}$, we have*

$$\mathbb{S}^n \wedge X \simeq \Sigma^n X$$

Proof. Immediate by induction on n. □

Using this we can immediately see that the $\Sigma \dashv \Omega$ adjunction (Lemma 2.3.1) is a special case of Lemma 2.4.1.

2.5 Eilenberg-MacLane Spaces

Recall that in classical homotopy theory a topological space X is an **Eilenberg-MacLane space** if exactly one of its homotopy groups is non-trivial. In particular, given $n \in \mathbb{N}$ and a group G (necessarily Abelian if $n \geq 2$), then X is an Eilenberg-MacLane space of type $K(G, n)$ if $\pi_n(X) = G$ and $\pi_i(X) = \mathbf{1}$ for all $0 < i \neq n$. Such spaces exist as CW-complexes for all n and G. Moreover, this space of type $K(G, n)$ is unique up to weak homotopy equivalence.

Daniel Licata and Eric Finster have given [LF14] the following construction of Eilenberg-MacLane spaces in homotopy type theory as higher inductive types.

Definition 2.5.1. Given a group G with identity element e and group operation $\odot$, $K(G,1)$ is defined to be the higher inductive type with the following constructors:

- A witness $\theta_{K(G,1)} : \mathit{is}\text{-}1\text{-}\mathit{type}(K(G,1))$
- A point $\mathit{base} : K(G,1)$
- A function $\mathit{loop} : G \to (\mathit{base} =_{K(G,1)} \mathit{base})$
- A function $\mathit{loop\text{-}ident} : \mathit{loop}(e) = \mathit{refl}_{\mathit{base}}$
- A function $\mathit{loop\text{-}comp} : \prod_{x,y:G} \mathit{loop}(x \odot y) = \mathit{loop}(x) \cdot \mathit{loop}(y)$

with recursion principle:

> Given a point $c : C$, a function $l : G \to (c =_C c)$, paths $p_e : l(e) = \mathit{refl}_c$ and $p_{x,y} : l(x \odot y) = l(x) \cdot l(y)$ for each $x, y : G$, and a witness $\theta : \mathit{is}\text{-}1\text{-}\mathit{type}(C)$, there is a function $f : K(G,1) \to C$ such that $f(\mathit{base}) \equiv c$ and $\mathit{ap}_f(\mathit{loop}(x)) = l(x)$ for all $x : G$. I.e. there is a function
>
> $$\mathit{rec}_G : \prod_{C:\mathcal{U}} \prod_{c:C} \prod_{l:G\to(c=c)} (l(e) =_C \mathit{refl}_c) \to \left(\prod_{x,y:G} (l(x \odot y) =_C l(x) \cdot l(y)) \right) \to \mathit{is}\text{-}1\text{-}\mathit{type}(C) \to (K(G,1) \to C)$$

Given $n \geq 2$, the type $K(G,n)$ is defined

$$K(G,n) :\equiv \left\| \Sigma^{n-1} K(G,1) \right\|_n$$

.

They prove that the type $K(G,n)$ defined above has the desired property that $\pi_n(X) \cong G$ and $\pi_i(X) = \mathbf{1}$ for all $0 < i \neq n$. In particular, for the case $n = 1$, they show that *loop* is an equivalence.

In fact, any $(n-1)$-connected type with these properties is equivalent to an Eilenberg-MacLane space, as proved by Floris van Doorn in his thesis [Doo18].

Lemma 2.5.1 ([Doo18] Theorem 4.2.4) - Uniqueness of Eilenberg-MacLane spaces. *Suppose (X, x_0) is a $(n-1)$-connected pointed n-type such that for some group G and $n : \mathbb{N}$, $\pi_n(X, x_0) \cong G$ and $\pi_i(X, x_0) = \mathbf{1}$ for all $0 < i \neq n$. Then there is an equivalence $X \simeq K(G, n)$.*

From this, and Lemma 2.1.3, we obtain

Corollary 2.5.1.1. $\Omega K(G, n+1) \simeq K(G, n)$

Example 2.5.1. We saw in Section 2.2 that $\mathbb{S}^1$ is a type with $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ and $\pi_i(\mathbb{S}^1) = \mathbf{1}$ for all $i > 1$. Moreover, $\mathbb{S}^1$ is 0-connected, so by Lemma 2.5.1, $\mathbb{S}^1 \simeq K(\mathbb{Z}, 1)$.

Chapter 3

Classification of Bundles

In this chapter we illustrate doing synthetic homotopy theory by introducing fibre bundles and principal bundles in homotopy type theory, before proving a series of results leading to the main theorem.

3.1 Fibre Bundles and Principal Bundles

We follow the presentation of fibre bundles given by Felix Wellen in [Wel17, Wel18]. He presents bundles from two viewpoints: as maps $p : E \to X$ from a total space to a base space; as well as a dependent version as families of types $E : X \to \mathcal{U}$.

Definition 3.1.1. Suppose $p : E \to X$ is a map of types. A map $w : W \to X$ **trivialises** p if w is surjective ($\prod_{x:X} \|fib_w(x)\|_{-1}$ is inhabited), and there is a pullback square

$$\begin{array}{ccc} W \times F & \longrightarrow & E \\ \downarrow{\scriptstyle pr_1} & & \downarrow{\scriptstyle p} \\ W & \xrightarrow{w} & X \end{array}$$

for some type F. In this case we say that $p : E \to X$ is an F**-fibre bundle**.

The idea here is that $w : W \to X$ represents an open cover of X. Then on any one part of the cover, the bundle looks like a product space. That is, a bundle *locally* looks like a product space, but may differ on a global level. Alternatively, from the dependent viewpoint we have

Definition 3.1.2. A map $E : X \to \mathcal{U}$ is trivialised by a surjective map $w : W \to X$ if for each $x : W$ we have

$$E(w(x)) \simeq F$$

for some type F. We say that $E : X \to \mathcal{U}$ is an F**-fibre bundle**.

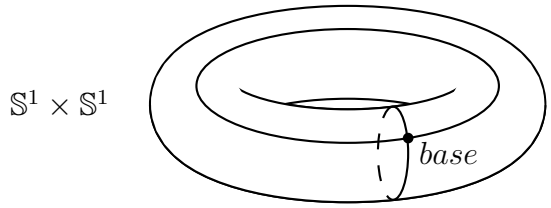


Figure 3.1: The torus: AKA the trivial $\mathbb{S}^1$-fibre bundle over $\mathbb{S}^1$.

Lemma 3.1.1 [Wel18] Theorem 4.10. *The above definitions are equivalent via the correspondence*

$$(p : E \to X) \mapsto (fib_p(x) : X \to \mathcal{U})$$
$$(E : X \to \mathcal{U}) \mapsto (pr_1 : (\sum_{x:X} E(x)) \to X).$$

The trivialisation condition also turns out to be equivalent to the condition that $\prod_{x:X} \|E(x) \simeq F\|_{-1}$ *is inhabited.*

Let's consider some examples from mathematics.

Example 3.1.1. Given X and F, the **trivial F-fibre bundle** over X is given by $pr_1 : X \times F \to X$. It is easily seen to be trivialised by the map $id_X : X \to X$. For example, the **torus** $\mathbb{S}^1 \times \mathbb{S}^1$, shown in Figure 3.1, is the trivial $\mathbb{S}^1$-fibre bundle over $\mathbb{S}^1$.

Example 3.1.2. As a non-trivial example we consider the **Klein bottle** as illustrated in Figure 3.1. Define a map $rev : \mathbb{S}^1 \to \mathbb{S}^1$ such that $rev(base) \equiv base$ and $ap_{rev}(loop) = loop^{-1}$ by circle induction. The map rev is clearly an equivalence (being quasi-inverse to itself). Hence we can define a map

$$code_K : \mathbb{S}^1 \to \mathcal{U}$$

such that $code_K(base) \equiv \mathbb{S}^1$ and $ap_{code_K}(loop) = ua(rev)$ by circle induction again. We call the total space $\sum_{x:\mathbb{S}^1} code_K(x)$ the Klein bottle, and we claim that $code_K$ expresses it as an $\mathbb{S}^1$-fibre bundle over $\mathbb{S}^1$: locally it looks like the torus $\mathbb{S}^1 \times \mathbb{S}^1$, but globally it's different. A rough sketch of how the trivialising pullback square might look is shown in Figure 3.3. There we split the circle into two halves, each equivalent to the interval (itself equivalent to the unit type, being contractible). This induces a split in the Klein bottle into two cylinders, which are the product of the circle with the intervals.

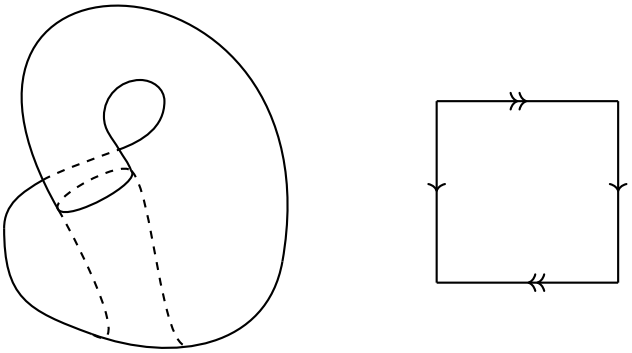


Figure 3.2: Left: An illustration of the Klein bottle immersed in $\mathbb{R}^3$. Right: Shown diagrammatically as a quotient of the square.

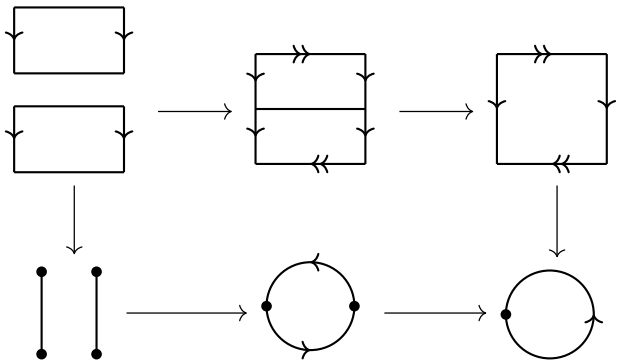


Figure 3.3: Trivialising pullback square for the Klein bottle.

Example 3.1.3. A **covering space** is a fibre bundle with discrete fibres. For example we expressed the universal cover of $\mathbb{S}^1$ as a map $code : \mathbb{S}^1 \to \mathcal{U}$ with fibre $\mathbb{Z}$. This is a $\mathbb{Z}$-fibre bundle over $\mathbb{S}^1$.

Definition 3.1.3. Given two bundles $p_1 : E \to X$ and $p_2 : D \to X$, a map $f : E \to D$ is a **bundle morphism** if $p_1 = p_2 \circ f$. That is f preserves the fibres. It is a **bundle isomorphism** if there is a map $g : D \to E$ in the other direction which is also a bundle morphism.

Before presenting the classification of fibre bundles by pullbacks and principal bundles, we will need some terminology from work by Ulrik Buchholtz, Floris van Doorn and Egbert Rijke on higher groups and group actions [BvDR18]. We will be focussing on the case of discrete groups (i.e. 0-types) where the definition of a group and of homomorphisms are familiar. But it is worth mentioning that much of the

following applies to groups with more homotopy structure and an interested reader should look at the original work for those definitions.

Definition 3.1.4. Given a group G, its **delooping** $(BG, base)$ is a pointed, connected type such that $\Omega(BG, base) \simeq G$.

Note that if G is a discrete group, then by uniqueness of Eilenberg-MacLane spaces (Lemma 2.5.1) we have that $BG \simeq K(G, 1)$.

Definition 3.1.5. Given $a : A$, the **automorphism** group of a is

$$Aut(a) :\equiv a =_A a.$$

It has a delooping

$$BAut(a) :\equiv \sum_{x:A} \|a = x\|_{-1}$$

with basepoint $(a, |refl_a|)$.

Definition 3.1.6. A **group action** of G on $a : A$ is a pointed map $ac : (BG, base) \to (A, a)$.

A group action is equivalently a group homomorphism $G \to Aut(a)$. However since we have not introduced the definition of a homomorphism in this context, we will just note that it corresponds to our usual notion when G is discrete.

Definition 3.1.7. If ac is a G-action on a type $A : \mathcal{U}$, then we can talk about its **homotopy quotient**:

$$ac//G :\equiv \sum_{z:BG} ac(z).$$

Returning to the theory of fibre bundles, we introduce the following *universal* F-fibre bundle. It is universal in the sense that all other F-fibre bundles will turn out to be pullbacks of this one.

Definition 3.1.8. Let $Aut(F)$ act on $F : \mathcal{U}$ via $(F', |\psi|) \mapsto F'$, and denote the homotopy quotient $\sum_{(F',|\psi|):BAut(F)} F'$ by $F//Aut(F)$. Then the map

$$pr_1 : F//Aut(F) \to BAut(F)$$

is called the **universal F-fibre bundle**.

Lemma 3.1.2 ([Wel17] Theorem 4.3.8). *A map $p : E \to X$ is an F-fibre bundle iff there is a map $\chi : B \to BAut(F)$ such that there is a pullback square*

$$\begin{array}{ccc} E & \longrightarrow & F//Aut(F) \\ \downarrow{\scriptstyle p} & & \downarrow{\scriptstyle pr_1} \\ X & \xrightarrow{\chi} & BAut(F) \end{array}$$

*In this case, χ is called the **classifying map** of p.*

We use this idea of producing bundles as pullbacks of a universal bundle to define principal bundles.

Definition 3.1.9. Let $Aut(F)$ act on $Aut(F)$ via $(F', |\psi|) \mapsto (F = F')$, and denote the homotopy quotient $\sum_{(F',|\psi|):BAut(F)}(F = F')$ by $EAut(F)$. Then the map

$$\pi :\equiv pr_1 : EAut(F) \to BAut(F)$$

is called the **universal principal $Aut(F)$-bundle**.

Note that $EAut(F)$ is clearly contractible, so we could instead have defined this as the map $\star \mapsto (F, |refl_F|)$. However, in the form presented above it is easier to check (as in [Wel17]) that this map is in fact surjective and an $Aut(F)$-fibre bundle. We now reach the final definition we want.

Definition 3.1.10. A map $p : E \to X$ is a **principal $Aut(F)$-bundle** if there is a pullback square

$$\begin{array}{ccc} E & \longrightarrow & EAut(F) \\ \downarrow{\scriptstyle p} & & \downarrow{\scriptstyle \pi} \\ X & \xrightarrow{\chi} & BAut(F) \end{array}$$

for some **classifying map** $\chi : X \to BAut(F)$. Principal bundle (iso)morphisms are defined as before.

Principal $Aut(F)$-bundles come equipped with an $Aut(F)$ action on the total space E which is free and transitive on the fibres. However it is beyond the scope of this work to see how to pull this action out from Definition 3.1.10. The following examples may give some insight though.

Example 3.1.4. Recall that a covering space $E : X \to 1\text{-}type$ is **regular** if the action of group of deck transformations $deck(E)$ on $\sum_{x:X} E(x)$ is transitive (it is always free). These are principal $deck(E)$-bundles.

Example 3.1.5. Ulrik Buchholtz and Egbert Rijke constructed the real projective spaces $\mathbb{R}P^n$ in homotopy type theory [BR17]. They also show that for each n there is a principal $\mathbb{Z}_2$-bundle $\mathbb{S}^n \to \mathbb{R}P^n$.

Wellen notes in particular that any principal $Aut(F)$-bundle p is automatically a surjective $Aut(F)$-fibre bundle.

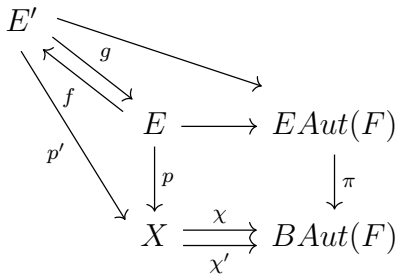
Lemma 3.1.3. *If $p : E \to X$ and $p' : E' \to X$ are both pullbacks of a map $\chi : X \to BAut(F)$ along π, then they are isomorphic as bundles.*

Proof. By the universal property of pullbacks we obtain maps $f : E \to E'$ and $g : E' \to E$ such that $p' \circ f = p$ and $p \circ g = p'$. Therefore f and g are bundle morphisms and the two bundles are isomorphic. □

We would also like isomorphic principal bundles to have equal classifying maps. This may be the case, but here only give a proof of a "propositional homotopy" between the two maps.

Lemma 3.1.4. *Suppose (E, p, χ, ϕ) and (E', p', χ', ϕ') are isomorphic principal $Aut(F)$-bundles over X. Then $\prod_{x:X} \|\chi(x) = \chi'(x)\|_{-1}$ is inhabited.*

Proof. Bundle isomorphism means there are maps $f : E \to E'$ and $g : E' \to E$ such that $p' \circ f = p$ and $p \circ g = p'$. Moreover, since $EAut(F)$ is contractible, so are the types $E \to EAut(F)$ and $E' \to EAut(F)$ by Lemma 1.5.2. Putting this together yields the following commutative diagram



In particular, $\chi \circ p = \chi' \circ p$ with witness ψ say. Now, as pointed out by Wellen, p is surjective, meaning in this case that $\prod_{x:X} \|fib_p(x)\|_{-1}$ is inhabited. By the universal property of truncations (Lemma 1.5.5), for each $x : X$, the map $((e, p_e) \mapsto |happly(\psi, e)|_{-1}) : fib_p(x) \to \|\chi(x) = \chi'(x)\|_{-1}$ extends to a map $\|fib_p(x)\|_{-1} \to \|\chi(x) = \chi'(x)\|_{-1}$. Therefore we can construct an inhabitant of $\prod_{x:X} \|\chi(x) = \chi'(x)\|_{-1}$. □

In the case where X is a set, we can apply a version of the **axiom of choice** which exists in homotopy type theory to obtain a proper homotopy, and hence an equality $\chi = \chi'$. However, this is not a restriction we are willing to put on X.

In any case, we see that understanding the function type $X \to BAut(F)$ is key to describing the principal $Aut(F)$-bundles we might get over the base space X.

3.2 Maps Between Eilenberg-MacLane Spaces

Recall that in the case where $Aut(F) \simeq G$ for some discrete group G, we have that $BAut(F)$ is simply $K(G,1)$. We therefore want to consider maps into $K(G,1)$, but its recursion principle only specifies how to construct maps out of it. However, looking at maps between Eilenberg-MacLane spaces gives a very useful lemma. The following proof by induction is outlined by Mike Shulman on the Homotopy Type Theory blog, but we will provide a more detailed account in the base case.

Lemma 3.2.1 [Shu14]. *Given $n : \mathbb{N}$ as well as discrete groups G and H, we have that $Map_*(K(G,n), K(H,n)) \simeq Hom(G,H)$, the type of group homomorphisms $G \to H$.*

Proof. We proceed by induction on $n \geq 1$.

Base Case $n = 1$ We will set up explicit maps between $Map_*(K(G,1), K(H,1))$ and $Hom(G,H)$ and show that they are quasi-inverse.

To define our function $F : Hom(G,H) \to Map_*(K(G,1), K(H,1))$ we recall the recursion principle for $K(G,1)$ given in Definition 2.5.1: There is a function

$$rec_G : \prod_{C:\mathcal{U}} \prod_{c:C} \prod_{l:G\to(c=c)} (l(e) =_C refl_c) \to \left(\prod_{x,y:G} (l(x \odot y) =_C l(x) \cdot l(y)) \right) \\ \to is\text{-}1\text{-}type(C) \to (K(G,1) \to C)$$

such that if $f \equiv rec_G(C, c, l, p_e, (x, y \mapsto p_{x,y}), \theta, -)$, then $f(base) \equiv c$ and $ap_f(loop(x)) = l(x)$ for each $x : G$. Given a group homomorphism $\phi : Hom(G,H)$, we therefore define

$$F(\phi) :\equiv rec_G(K(H,1),\, base_{K(H,1)},\, loop_{K(H,1)} \circ \phi,\, loop\text{-}ident_{K(H,1)}, \\ loop\text{-}comp_{K(H,1)},\, \theta_{K(G,1)},\, -)$$

noting that $loop\text{-}ident_{K(H,1)}$ and $loop\text{-}comp_{K(H,1)}$ give the right paths since ϕ is a homomorphism and H is a set.

To define our function $G : Map_*(K(G,1), K(H,1)) \to Hom(G,H)$ we will make use of an *encode* style argument. First note that if $x : H$ then since group elements are invertible, the function $f_x : h \mapsto x \odot h$ is an equivalence $H \simeq H$. Therefore, by univalence constrained to the universe 0-*type*, there is a path $ua(f_x) : H =_{0\text{-}type} H$. Since, by 0-*type* is a 1-type by Lemma 1.5.3, the recursion principle for Eilenberg-MacLane spaces lets us construct a map $code : K(H,1) \to 0\text{-}type$ such that $code(base) \equiv H$ and $ap_{code}(loop(x)) = ua(f_x)$ for each $x : H$. We now define $encode : \prod_{z:K(H,1)}(base =_{K(H,1)} z) \to code(z)$ as $encode_z(p) :\equiv transport^{code}(p,e)$. But now we see that for all $x : H$

$$\begin{aligned}
transport^{code}(loop(x), e) &= transport^{A\mapsto A}(ap_{code}(loop(x)), e) && \text{(by Lemma 1.3.8)}\\
&= transport^{A\mapsto A}(ua(f_x), e) && \text{(by definition of } code\text{)}\\
&= idtoeqv(ua(f_x))(e) && \text{(by definition of } idtoeqv\text{)}\\
&= f_x(e) && \text{(by univalence)}\\
&= x
\end{aligned}$$

So in a sense, $encode_{base} : (base =_{K(H,1)} base) \to H$ actually *decodes* the group out of the construction of $K(H,1)$ (but we call it *encode* to maintain the parallel with the example of computing $\pi_1(\mathbb{S}^1)$ in Section 2.2). Finally, given $\sigma : Map_*(K(G,1), K(H,1))$, we define

$$G(\sigma) :\equiv encode_{base} \circ ap_\sigma \circ loop_{K(G,1)}$$

That is, given $g : G$, we first take it to the corresponding loop in $K(G,1)$, use σ to map this to a loop in $K(H,1)$, and then pull out the corresponding element of H. It's easy to check that this is a well defined group homomorphism.

It now remains to check that F and G are quasi-inverse. However, we see that given $\phi : Hom(G,H)$, then for each $x : G$ we have

$$\begin{aligned}
G(F(\phi))(x) &\equiv encode_{base}(ap_{F(\phi)}(loop_{K(G,1)}(x))) && \text{(by definition of } G\text{)}\\
&= encode_{base}(loop_{K(H,1)}(\phi(x))) && \text{(by definition of } F\text{)}\\
&= \phi(x) && \text{(by the calculation above)}
\end{aligned}$$

so that by function extensionality we have $G(F(\phi)) = \phi$. The other direction is slightly more tricky. One way of showing it is by making use of the induction principle for $K(G,1)$. This says that, given some family $C : K(G,1) \to 1\text{-}type$, along with $c : C(base)$ and a family of maps $p : \prod_{x:G} c =^C_{loop(x)} c$ which preserves identity and

composition, then there is a function $f : C$ with the obvious properties. We define $C :\equiv \lambda z.(FG\sigma(z) = \sigma(z))$ and let c be the path $FG\sigma(base) \equiv base = \sigma(base)$ given by the definition of F and the fact that σ is a pointed map. The subtle bit is noticing that $ap_C(loop(x)) : ((FG\sigma(base) = \sigma(base)) = (FG\sigma(base) = \sigma(base)))$ and so, since $K(H, 1)$ is a 1-type, we have $ap_C(loop(x)) = refl_{FG\sigma(base)=\sigma(base)}$. Now for each $x : G$ we can give the path

$$\begin{aligned}
&transport^{z\mapsto FG\sigma(z)=\sigma(z)}(loop(x), c) \\
&\qquad = transport^{A\mapsto A}(ap_{z\mapsto FG\sigma(z)=\sigma(z)}(loop(x)), c) && \text{(by Lemma 1.3.8)} \\
&\qquad = transport^{A\mapsto A}(refl_{FG\sigma(base)=\sigma(base)}, c) && \text{(since } K(H,1) \text{ is a 1-type)} \\
&\qquad = id_{FG\sigma(base)=\sigma(base)}(c) && \text{(by definition of } transport) \\
&\qquad = c
\end{aligned}$$

This will respect the identity and composition, hence by induction we have $f : C \equiv \prod_{z:K(G,1)}(FG\sigma(z) = \sigma(z))$, but now function extensionality therefore gives us that $FG(\sigma) = \sigma$.

Induction Step By Definition 2.5.1 we have that $K(G, n + 1) = \|\Sigma K(G, n)\|_{n+1}$. Therefore, since $K(H, n + 1)$ is an $(n + 1)$-type,

$$Map_*(K(G, n + 1), K(H, n + 1)) \simeq Map_*(\Sigma K(G, n), K(H, n + 1))$$

by the universal property of truncations (Lemma 1.5.5). Now applying the suspension-loop adjunction (Lemma 2.3.1), we have

$$Map_*(\Sigma K(G, n), K(H, n + 1)) \simeq Map_*(K(G, n), \Omega K(H, n + 1)).$$

But now by Corollary 2.5.1.1, $\Omega K(H, n+1) = K(H, n)$. So by the induction hypothesis, $Map_*(K(G, n), \Omega K(H, n + 1))$ is $Hom(G, H)$.

□

Corollary 3.2.1.1.

$$Map_*(\mathbb{S}^1, \mathbb{S}^1) \simeq \mathbb{Z}$$

Proof. In Example 2.5.1 we saw that $\mathbb{S}^1$ is $K(\mathbb{Z}, 1)$. Hence, by Lemma 3.2.1 we have

$$Map_*(\mathbb{S}^1, \mathbb{S}^1) \simeq Hom(\mathbb{Z}, \mathbb{Z}) \simeq \mathbb{Z}$$

(since a group homomorphism $\mathbb{Z} \to \mathbb{Z}$ is determined by where it sends 1). □

3.3 Classifying Principal Bundles of Discrete Groups

We're now ready to tackle the main theorem. We first note that, as mentioned in Section 3.1, when G is a discrete group we have that $BG \simeq K(G,1)$. Moreover $Aut(base : K(G,1)) \cong G$. Suppose our base space is X. Then according to Definition 3.1.10, we are therefore interested in maps $X \to K(G,1)$. In particular, we prove a result corresponding to a classical theorem of Gottlieb. The statement of the original theorem is

Theorem 3.3.1 ([Got69] Lemma 2). *Let X be a connected CW complex, G a discrete group, and $f : X \to K(G,1)$ be a pointed map. Then $\pi_1 Map(X, K(G,1); f)$ is isomorphic to $C_G(im\pi_1 f)$, where $C_G(S)$ is the centraliser of $S \subseteq G$.*

Note in particular, that while f is pointed, the mapping space involved is not. The idea behind my proof of this in homotopy type theory is based on the classical proof outlined by Tsutaya in [Tsu15] as Theorem 4.4. He considers the long exact sequence in homotopy coming from the evaluation map $ev_{x_0} : Map(X, K(G,1)) \to K(G,1)$ where $ev_{x_0}(g) :\equiv g(x_0)$ and x_0 is the basepoint of X preserved by f. Note that the fibre of this map over $base : K(G,1)$ is just the type of pointed maps $Map_*(X, K(G,1))$. For this proof we will require two lemmas:

Lemma 3.3.2. *If (X, x_0) is a pointed, 0-connected type, then*

$$\|Map_*(X, K(G,1))\|_0 \simeq Hom(\pi_1 X, G)$$

Proof. This follows fairly simply from Lemma 3.2.1, since

$$\begin{aligned} Map_*(X, K(G,1)) &\simeq Map_*(\|X\|_1, K(G,1)) && \text{by Lemma 1.5.5} \\ &\simeq Map_*(K(\pi_1 X, 1), K(G,1)) && \text{by Lemma 2.5.1} \\ &\simeq Hom(\pi_1 X, G) && \text{by Lemma 3.2.1} \end{aligned}$$

where we've been able to apply the uniqueness of Eilenberg-MacLane spaces since X is 0-connected. But now, $Hom(\pi_1 X, G)$ is a 0-type already, so this equivalence holds for the truncation. We also note at this point that this equivalence essentially sends a map $f : X \to K(G,1)$ to the induced map $\pi_1 f$.

□

Lemma 3.3.3. *If (X, x_0) is a 0-connected pointed type (path-connected space), and G is a 0-truncated (discrete) group, then $\pi_i(Map_*(X, K(G,1)), f)$ is trivial for all $i \geq 1$.*

Proof. We have the following sequence of equivalences (omitting writing the basepoints)

$$\begin{aligned}
\Omega^n Map_*(X, K(G,1)) &\simeq Map_*(\mathbb{S}^0, \Omega^n Map_*(X, K(G,1))) && \text{as } \mathbb{S}^0 \simeq \mathbf{2} \\
&\simeq Map_*(\mathbb{S}^n, Map_*(X, K(G,1))) && \text{by Lemma 2.3.1} \\
&\simeq Map_*(\mathbb{S}^n \wedge X, K(G,1)) && \text{by Lemma 2.4.1} \\
&\simeq Map_*(\Sigma^n X, K(G,1)) && \text{by Lemma 2.4.2.1}
\end{aligned}$$

Now if $n \geq 2$, we immediately see that

$$\begin{aligned}
Map_*(\Sigma^n X, K(G,1)) &\simeq Map_*(\Sigma^{n-2} X, \Omega^2 K(G,1)) && \text{by Lemma 2.3.1} \\
&\simeq Map_*(\Sigma^{n-2} X, \mathbf{1}) && \text{by construction} \\
&\simeq \mathbf{1}
\end{aligned}$$

Otherwise if $n = 1$, we use the fact that X is 0-connected and G is discrete

$$\begin{aligned}
Map_*(\Sigma X, K(G,1)) &\simeq Map_*(X, \Omega K(G,1)) && \text{by Lemma 2.3.1} \\
&\simeq Map_*(X, G) && \text{by construction}
\end{aligned}$$

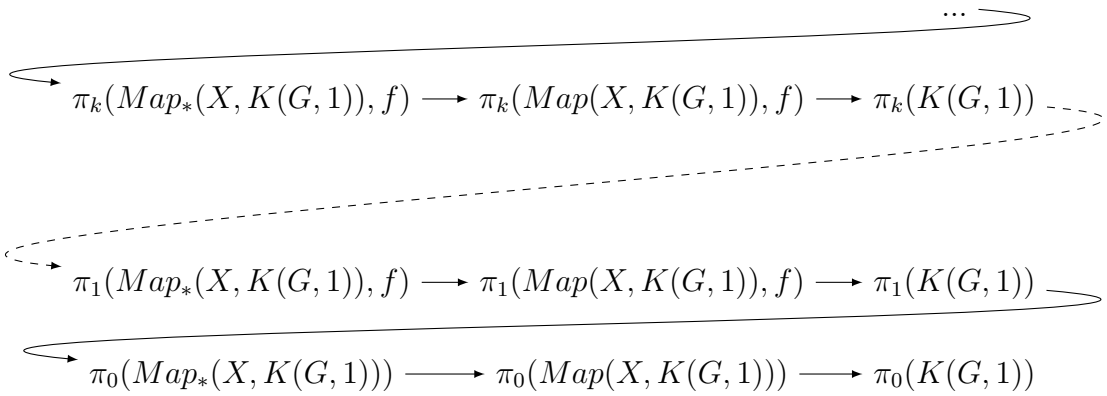
Since X is 0-connected, $\|x = x_0\|_{-1}$ is inhabited for all $x : X$. But now since G is a set, given any $f : Map_*(X, G)$, the type $(f(x) = f(x_0))$ is a (-1)-type. Moreover, $f(x_0) = e$ (in fact $f(x_0) \equiv e$). Hence the universal property of truncations (Lemma 1.5.5) allows us to define a map $\prod_{x:X}(f(x) = e)$. In particular, every map in $Map_*(X, G)$ is homotopic to the constant map $x \mapsto e$. Hence $Map_*(X, G)$ is contractible, and equivalent to $\mathbf{1}$ by Lemma 1.5.1. □

Writing $Map(X, Y; f)$ for $\Sigma_{g:Map(X,Y)}(g = f)$ (i.e. the path component of $Map(X, Y)$ containing f), we are now ready to prove

Theorem 3.3.4. *Let (X, x_0) be a pointed 0-connected type, G a 0-truncated (discrete) group, and $f : X \to K(G,1)$ a pointed map. Then*

$$Map(X, K(G,1); f) \simeq K(C_G(im\pi_1 f), 1)$$

Proof. Consider the map $ev_{x_0} : Map(X, K(G,1)) \to K(G,1)$ which sends $g \mapsto g(x_0)$. The fibre over $base : K(G,1)$ is $fib_{ev_{x_0}}(base) \equiv \Sigma_{g:Map(X,K(G,1))}(ev_{x_0}(g) = base) \simeq \Sigma_{g:Map(X,K(G,1))}(g(x_0) = base) \equiv Map_*(X, K(G,1))$. Giving the mapping spaces basepoint f (in particular, let $\phi_0 : f(x_0) \equiv ev_{x_0}(f) = base$) and applying the long exact sequence in homotopy of a fibration (Lemma 2.1.2) yields the exact sequence



Noting that $\pi_k(K(G,1)) \simeq \mathbf{1}$ for all $k \geq 2$ and applying Lemma 3.3.3, we immediately see by exactness that $\pi_k(Map(X, K(G,1)), f) \simeq \mathbf{1}$ for all $k \geq 2$. It remains to calculate $\pi_1(Map(X, K(G,1)), f)$.

Applying Lemma 3.3.2 and substituting into the sequence above gives an exact sequence

$$\mathbf{1} \to \pi_1(Map(X, K(G,1)), f) \to G \xrightarrow{\partial} Hom(\pi_1 X, G)$$

Now, from Lemmas 2.1.2, 3.3.2, and the construction of $K(G,1)$, we see that ∂ sends $g \mapsto (|loop(g)|_0 : \pi_1 K(G,1)) \mapsto ((f, \phi_0 \cdot loop(g)) : Map_*(X, K(G,1))) \mapsto (\alpha_g \circ \pi_1 f : Hom(\pi_1 X, G))$, where $\alpha_g : G \to G$ is the map $h \mapsto g^{-1} \odot h \odot g$. By exactness we immediately see that $\pi_1(Map(X, K(G,1)), f) \simeq ker(\partial) \simeq \{g : G \mid \alpha_g \circ \pi_1 f = \pi_1 f\} \simeq C_G(im \pi_1 f)$.

Since $Map(X, K(G,1))$ is a 1-type by Lemma 1.5.2, so is $Map(X, K(G,1); f)$. Hence we can appeal to the uniqueness of Eilenberg-MacLane spaces (Lemma 2.5.1) to give the desired equivalence. □

Corollary 3.3.4.1. *Let G be a discrete group. Then*

$$Map(K(G,1), K(G,1); id) \simeq K(Z(G), 1)$$

where $Z(G)$ is the centre of G.

To end, we will mention that this result immediately says something about first cohomology groups. It is outside the scope of this work to introduce cohomology, but in his thesis [Cav15], Evan Cavallo mentions that we can define the **ordinary cohomology** for $n \geq 0$ via $H^n(X) :\equiv \|Map(X, K(G,n))\|_0$. Our result therefore immediately tells us about $H^1(X)$.

References

[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 146, pages 45–55, 2009.

[BR17] Ulrik Buchholtz and Egbert Rijke. The real projective spaces in homotopy type theory. *Proceedings - Symposium on Logic in Computer Science*, pages 1–8, 2017.

[Bru16] Guillaume Brunerie. *On the homotopy groups of spheres in homotopy type theory*. PhD thesis, Nice Sophia Antipolis University, 2016.

[BvDR18] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. Higher groups in homotopy type theory. *Preprint arXiv:1802.04315*, 2018.

[Cav15] Evan Cavallo. Synthetic cohomology in homotopy type theory. Master's thesis, Carnegie Mellon University, 2015.

[Doo18] Floris Van Doorn. *On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory*. PhD thesis, Carnegie Mellon University, 2018.

[Fra17] Andreas Franz. The wedge sum and the smash product in homotopy type theory. Master's thesis, LMU Munich, 2017.

[Got69] Daniel H. Gottlieb. Covering transformations and universal fibrations. *Illinois Journal of Mathematics*, 13(2):432–437, 1969.

[INR] INRIA. The coq proof assistant. `https://coq.inria.fr/`.

[LF14] Daniel R. Licata and Eric Finster. Eilenberg-maclane spaces in homotopy type theory. *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the*

Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) - CSL-LICS '14, pages 1–9, 2014.

[LS13] Daniel R. Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *Proceedings - Symposium on Logic in Computer Science*, pages 223–232, 2013.

[ML75] Per Martin-Löf. An intuitionistic theory of types: predicative part. In *Logic Colloquium '73, Proceedings of the Logic Colloquium*, volume 80, pages 73–118. North-Holland, 1975.

[Nor] Ulf Norell. Agda. `https://wiki.portal.chalmers.se/agda/pmwiki.php`.

[Shu14] Mike Shulman. Blog article: Fibrations with fibre an eilenberg-maclane space. Homotopy Type Theory Blog `https://homotopytypetheory.org/2014/06/30/fibrations-with-em-fiber/`, 2014. Accessed: 22-01-2019.

[Tsu15] Mitsunobu Tsutaya. Lecture notes: Evaluation fiber sequence and homotopy commutativity. `www2.math.kyushu-u.ac.jp/~tsutaya/lecture_evaluation_20150724.pdf`, 2015.

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

[Voe06] Vladimir Voevodsky. A very short note on the homotopy λ-calculus. `http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf`, 2006.

[Wel17] Felix Wellen. *Formalizing Cartan Geometry in Modal Homotopy Type Theory*. PhD thesis, Karlsruher Institute of Technology, 2017.

[Wel18] Felix Wellen. Cartan geometry in modal homotopy type theory. *Preprint, arXiv:1806.05966*, 2018.