Lecture Notes on Synthetic Homotopy Theory

Hugo Moeneclaey

Abstract

These are lecture notes for an introductory class given at LMFI in 2019-2020. They were given together with exercise sheets.

Contents

1 Introduction

1.1 A quick introduction to fundamental mathematics

In fundamental mathematics one is interested in some class of objects.

- A group theorist is interested in groups.
- A topologist is interested in topological spaces.
- A geometer is interested in smooth manifolds (often with additional structure).
- A number theorist is interested in number fields.
- An algebraic geometer is interested in schemes.
- ...

A fundamental insight (generally attributed to Grothendieck) is that one should be interested in morphisms between objects as well.

- A group theorist is interested in morphisms of groups.
- A topologist is interested in continuous maps.
- A geometer is interested in smooth maps (possibly preserving the additional structure).
- ...

Moreover some morphisms are called *isomorphisms*, and isomorphic objects should be considered the same.

- A morphism of group is an isomorphism if it is a bijection.
- A morphism of space is an isomorphism if it is continuous bijection with a continuous inverse. This is called an homeomorphism.
- A morphism of manifold is an isomorphism if it is a smooth bijection with a smooth inverse. This is called a diffeomorphism.
- ...

This heuristic "isomorphic objects are the same" is often used nowadays. For example if you want to define a meaningful new property of groups, the first thing you should do is to check that it is invariant by isomorphisms. This point of view is emphasized by *category theory*. Some category theorists even say that a notion not invariant under isomorphisms is "evil". "Evil" notions are sometimes useful, but we are rarely interested in them per se.

Remark 1. *We will see that in homotopy type theory this heuristic is actually an axiom, called the* univalence axiom.

In this situation it is natural to try to understand objects up to isomorphisms. The "holy grail" is to give a classification, generally by giving:

- A list L of non-isomorphic objects, such that any object is isomorphic to one of them.
- Some kind of procedure to tell which object of L is isomorphic to a given object.

This is called a *classification*. This is usually done by defining *invariants*, i.e. things (e.g. natural numbers) which are not changed under isomorphisms. Having a full classification is usually extremely hard, and often impossible. But invariants are nevertheless very useful, as they allow us to tell objects apart.

Remark 2. *We list some example of classification results linked to groups. Note that a classification of finitely generated groups is impossible (essentially because the word problem is undecidable).*

- *Finitely generated abelian groups have been classified. This means they are not generally considered a research subject, while being extremely useful in mathematics.*
- *Countably generated abelian groups have not been classified. So they are a research subject, and not as useful as finitely generated ones.*
- *Finite simple groups have been classified. This is generally considered as one of the great mathematical achievements.*
- *Classifying finite groups is an extremely difficult problem.*

Of course there is a lot more to fundamental mathematics than this, but in my opinion it is a useful picture to have in mind when discovering a new subject.

1.2 Algebraic topology and homotopy theory

I introduce the fundamental notion of homotopy, in the usual mathematical framework.

Definition 1. *Assume given $f, g : X \to Y$ two continuous maps between topological spaces. A homotopy from f to g is a continuous map:*

$$H : X \times [0,1] \to Y$$

where $[0,1]$ is the real interval with its euclidian topology, such that:

$$H(x,0) = f(x)$$

$$H(x,1) = g(x)$$

Intuitively this is a path from f to g in the space of maps from X to Y. One might consider maps between spaces up to homotopy, and this motivates the following definition:

Definition 2. *A continuous map $f : X \to Y$ is called a homotopy equivalence if there exists a continuous $g : Y \to X$ such that $f \circ g$ and $g \circ f$ are homotopic to the identity maps.*

In the 1920s some people were investigating the classification of topological spaces. It turns out most of the useful invariants they came up with were invariant not only under homeomorphisms, but also under homotopy equivalences. This leads to algebraic topology, which studies invariants under homotopy equivalences. The most well-known invariant of algebraic topology are groups (usually finitely generated, and often abelian), they are called homotopy, homology and cohomology groups.

After some time, people started to be interested in *homotopy types*, which are defined as spaces up to homotopy equivalence. In fact they occur more often and more naturally than topological spaces in mathematics. I call the study of homotopy types *homotopy theory*, although this is not a universal terminology.

Remark 3. *I list some examples of situations where a homotopy type occurs. This is indicative, and you are not supposed to understand any of them.*

- *From a presentation of a group it is possible to build a space. Two presentations of the same group will give homotopy equivalent spaces, which are not homeomorphic in general.*
- *A groupoid is a category where all morphisms are invertible. From a groupoid it is possible to build a space, and equivalent groupoids give non-homeomorphic but homotopy equivalent spaces.*
- *It is possible to build a space from chain complexes, and quasi-isomorphic chain complexes give non-homeomorphic but homotopy equivalent spaces.*

1.3 ∞-groupoids

The notion of topological space is not very convenient to work with so that nowadays mathematicians use other presentations of homotopy types. We now sketch one such presentation using ∞-groupoids. The key idea is that when we consider a space up to homotopy equivalence we can forget its topology and remember only about its points, its paths between points, its homotopies between paths, etc...

First we introduce groupoids as a pedagogical example. A groupoid is defined as a category where all morphisms are invertible. This means that a groupoid consists of:

- A set whose element are called points of the groupoid.
- For any two points x, y a set called the set of paths from x to y.
- For any point x a path e_x from x to x called the unit (or identity) of x.

- For any paths p from x to y and q from y to z, a path $p \cdot q$ from x to z called the multiplication (or composition) of p and q.
- For any p from x to y a path p^{-1} from y to x called the inverse of p.

Such that e, $\_\cdot\_$ and $\_^{-1}$ obey the laws of a group. There is a geometrical intuition, which should be transparent from our choice of names for the elements. From a topological space X it is possible to build a groupoid $\Pi_1(X)$ called its fundamental groupoid. The points of $\Pi_1(X)$ are the points of X, and the paths of $\Pi_1(X)$ are the equivalence classes of paths in the X up to homotopy.

Exercise 1. *Complete the definition of the fundamental groupoid of a topological space. Convince yourself that it is necessary to consider paths up to homotopy, rather than just paths (hint: some group laws fail, can you tell which ones ?).*

There is a notion of equivalence of groupoids (actually defined as an equivalence of categories), such that a homotopy equivalence between spaces induces an equivalence between their fundamental groupoids. But we loose some information by going from a space to its fundamental groupoid, even when considering groupoids up to equivalences. In fact some non-homotopy equivalent spaces have equivalent fundamental groupoids. This is due to the fact that we brutally quotiented by homotopy between paths, loosing the structure of dimension higher than one in the space.

The notion of ∞-groupoid corrects this defect. An ∞-groupoid consists of:

- A set of points.
- For any two points a set of paths between them.
- For any two paths between two points, a set of homotopies between them.
- $\dots$

Moreover it should have a lot of compositions, inverses and units, for example multiplications of paths and homotopies, which should obey a lot of rules *up to homotopy*. This means for example that for composable paths p, q, r, we should have multiplications $(p \cdot q) \cdot r$ and $p \cdot (q \cdot r)$, as well as a homotopy between them. Listing all such rules is quite technical, and is usually done using the so-called Kan simplicial sets. But there are a lot of other definitions of ∞-groupoids.

There is a criteria for a notion of ∞-groupoids to be valid. From a space X should be possible to build its *fundamental ∞-groupoid* $\Pi_\infty(X)$. Moreover there should be a notion of equivalence of ∞-groupoids such that homotopy equivalences between spaces induce equivalences between their fundamental ∞-groupoids. Finally we should have the following result:

Homotopy types correspond precisely to ∞-groupoids up to equivalences.

This is called the *homotopy hypothesis*. It means that all notions of ∞-groupoids are equivalent, and that we can use ∞-groupoids rather than topological spaces to represent homotopy types.

1.4 Homotopy type theory

Now we talk about a seemingly unrelated story. In the 1970s Martin-Löf introduced a formal system called type theory. His goal was to provide a foundation for constructive mathematics. The main objects of type theory are types and their inhabitants, with types playing the role of sets and proposition, and inhabitants playing the role of elements of sets and proofs of propositions. This merging of elements and proofs means that type theory is *proof-relevant*, i.e. it is meaningful to ask whether two proofs of a propositions are the same.

The word "type" in type theory is related to the notion of type in computer science. Indeed we can see a type as the specification of the behavior of its inhabitants, which are seen as programs. As an example consider the proposition:

"for all natural numbers m and $n \neq 0$, there exists natural numbers q and $r < n$ such that $m = qn + r$"

In type theory it corresponds to a type whose inhabitants are programs computing the euclidian division. So type theory is both a foundational system for mathematics and a programming language.

There has been some debate concerning equality in type theory, especially about proof-relevance. We explain the problem. Assume we have two proofs p and q that $a = b$, i.e. we have $p, q : a = b$. Can we assert that $p = q$? At first sight it seems more natural to assume this. Indeed if we have two such proofs, and we have $s : P(a)$ an inhabitant of some type depending on a. Then we can build two inhabitants in $P(b)$, one using p and the other using q. If we do not assume that $p = q$, we can not assert that these two inhabitants are equal, contradicting the usual mathematical practice. So for some time this equality $p = q$ was assumed one way or the other when using type theory, so that it stays closer to usual mathematics.

In 1998 Martin Hofmann and Thomas Streicher gave a model of type theory where this rule $p = q$ does not hold [13]. In this model types were interpreted as groupoids, and proofs of equalities were interpreted as morphisms in the groupoid. In 2006, Steve Awodey and his student Michael Warren presented the first higher dimensional model of type theory. In 2009 Voevodsky worked out the details of an interpretation of types from type theory as ∞-groupoids [15], and therefore as homotopy types. This result forms the basis of *Homotopy Type Theory*.

Remark 4. *This is a lucky coincidence that types from type theory are related to types from homotopy theory, as both terms have distinct origins. The name "Homotopy Type Theory" is a play on words between "homotopy type" and "type theory", coined by Steve Awodey.*

In this class we will mainly investigate one consequence of this:

If we prove something about types in type theory, then this fact holds true for homotopy types.

This method of proving facts about homotopy types is called *synthetic homotopy theory*. It is useful because types (even with a proof-relevant equality) are often easier to manipulate than homotopy types.

This interpretation validates a new axiom called univalence, roughly stating that being isomorphic and being equal is the same thing for types. This principle dear to category theorists has a lot of interesting consequences. So we will do some maths in type theory with univalence, keeping in mind that it gives interesting result in homotopy theory through the interpretation. We will see that univalence indeed imply homotopy theoretic behavior for types, so that we have the following slogan:

> The principle that isomorphic objects are equal can be taken seriously, and it implies objects have an homotopy theoretic behavior.

Remark 5. *We will not give any formal account of the interpretation, although we will give some intuition about it through examples.*

2 Functions, products and identities

Here we give a quick introduction to the constructors of type theory, mainly to agree on notations. We sketch their interpretation as homotopy types or ∞-groupoids.

2.1 Universe

Assumption 1. *There is a type $\mathcal{U}$ whose inhabitants are types.*

This is definitely vague, for us it means that "let A be a type" would be formalized as "let A be of type $\mathcal{U}$", and that for $A : \mathcal{U}$, it makes sense to consider $x : A$.

Remark 6. *The rule $\mathcal{U} : \mathcal{U}$ leads to inconsistency (intuitively this is similar to Russell paradox: we cannot assume that the collection of all sets is a member of itself). One can introduce a second universe $\mathcal{U}_1$, which has the same rule as $\mathcal{U}$ plus the rule $\mathcal{U} : \mathcal{U}_1$. In most proof assistants based on type theory (Coq, Agda...), one assumes a whole hierarchy of universes $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$, so that any universe is in some other "larger" universe.*

The interpretation of $\mathcal{U}$ as an ∞-groupoid has ∞-groupoids as points, and equivalences as paths between them. Note that the interpretation of $\mathcal{U}$ as a topological spaces is rather unclear, justifying the need to introduce a more flexible notion of ∞-groupoid.

2.2 Function types

Assumption 2. *Assume given two types $A, B : \mathcal{U}$.*

- *(Formation) We have a type $A \to B$ in $\mathcal{U}$.*
- *(Elimination) From $f : A \to B$ and $a : A$, we can build $f(a) : B$.*
- *(Introduction) We can define a function $f : A \to B$ by defining $f(x) :\equiv t$ with t of type B assuming $x : A$.*

The function defined by $f(x) :\equiv t$ is denoted by $\lambda(x : A).\, t$ (in traditional mathematical notation this would be denoted by $x \mapsto t$). Then we have the definitional equality:

$$(\lambda(x : A).\, t)(a) \equiv t[x := a]$$

The type $A \to B$ is interpreted as the homotopy type of continuous maps from A to B, or equivalently the ∞-groupoid of morphisms of ∞-groupoids from A to B.

Remark 7. *An important example is the type $A \to \mathcal{U}$ for $A : \mathcal{U}$, which is seen as the type of families of types indexed by A. An element of this type is interpreted as the data of an ∞-groupoid for each point in A, an equivalence between them for each path in A, and so on...*

2.3 Dependent functions

Assumption 3. *Assume given a type $A : \mathcal{U}$ and a family of types $B : A \to \mathcal{U}$.*

- *(Formation) We have a type $(x : A) \to B(x)$.*
- *(Elimination) From $f : (x : A) \to B(x)$ and $a : A$, we can build $f(a) : B(a)$.*
- *(Introduction) We can define a function $f : (x : A) \to B(x)$ by defining $f(x) :\equiv t$ with t of type $B(x)$ assuming $x : A$.*

As for function types, the dependent function defined by $f(x) :\equiv t$ is denoted by $\lambda(x : A).\, t$. Notations seem to conflict with function types, but this is not a problem because $A \to B$ can be defined as $(x : A) \to \tilde{B}(x)$ where $\tilde{B}$ is the constant family in $A \to \mathcal{U}$ defined by $\tilde{B}(x) :\equiv B$.

We introduce some notations for identity morphisms and compositions, i.e. we define $\mathbf{id}_A :\equiv \lambda(x : A).x$ and $g \circ f :\equiv \lambda(x : A).\, g(f(x))$.

We do not make explicit the interpretation of $(x : A) \to B(x)$ as an ∞-groupoid.

2.4 The identity types

Inductive Definition 1. *Assume given a type A in $\mathcal{U}$.*

- *(Formation) For all $x, y : A$, then we have a type $x =_A y$ in $\mathcal{U}$.*
- *(Introduction) For all $x : A$, we have:*

$$\mathrm{refl}_x : x =_A x$$

- *(Elimination) In order to define a dependent function:*

$$f : (x, y : A) \to (p : x =_A y) \to P(x, y, p)$$

it is enough to define for all $x : A$:

$$f(x, x, \mathrm{refl}_x) :\equiv t$$

with $t : P(x, x, \mathrm{refl})$.

The elimination principle of identity types is called path induction. So, for example we will often say "by path induction, we may assume that $p : x =_A y$ is refl_x". We will often omit the subscript A in $x =_A y$ and simply write $x = y$.

⚠**Warning 1.** *Path induction is only valid when x and y are distinct variables, for example we can not assume that $p : f(x) =_B f(y)$ or $p : x =_A x$ are equal to* refl.

For A a type and $x, y : A$, the type $x = y$ is interpreted as the homotopy type of paths from x to y in A.

Remark 8. *We give a geometric interpretation of path induction. Assume we want to prove something for all $x, y : A$ and $p : x =_A y$. This is interpreted as a path in A with free endpoints, but we see intuitively that such a path can be deformed to a constant one, i.e. a reflexivity. So it is enough to prove what we want for reflexivities. Note that a path with only one free endpoint can also be deformed to a reflexivity, suggesting the next assumption.*

Next principle is very useful. It follows from the usual path induction principle, but we will not prove this. For us (and for example Agda) it can be understood as an alternative definition of identity types.

Assumption 4. *Assume given $a : A$. In order to define a dependent function:*

$$f(x : A) \to (p : x =_A a) \to C(x, p)$$

it is enough to define:

$$f(\mathrm{refl}_a) :\equiv t$$

with $t : C(a, \mathrm{refl}_a)$.

This assumption is called based path induction.

2.5 Product types

Inductive Definition 2. *Assume given two types $A, B : \mathcal{U}$.*

- *(Formation) We have a type $A \times B$ in $\mathcal{U}$*
- *(Introduction) For all $x : A$ and $y : B$ we have:*

$$(x, y) : A \times B$$

- *(Elimination) In order to define a dependent function:*

$$f : (z : A \times B) \to P(z)$$

it is enough to define for all $x : A$ and $y : B$:

$$f(x, y) :\equiv t$$

with $t : P(x, y)$.

Note that $f(x,y)$ is a shorthand for $f((x,y))$. We can define:

$$p_1 : A \times B \to A$$

$$p_2 : A \times B \to B$$

bu $p_1(x,y) :\equiv x$ and $p_2(x,y) :\equiv y$.

The interpretation of $A \times B$ as an ∞-groupoid has pairs of points in A and B as points, pairs of paths as paths, etc...

2.6 Dependent product types

Inductive Definition 3. *Assume given a type $A : \mathcal{U}$ and a family of types $B : A \to \mathcal{U}$.*

- *(Formation) We have a type $(x : A) \times B(x)$ in $\mathcal{U}$*
- *(Introduction) For all $x : A$ and $y : B(x)$ we have:*

$$(x,y) : (x : A) \times B(x)$$

- *(Elimination) In order to define a dependent function:*

$$f : (z : (x : A) \times B(x)) \to P(z)$$

it is enough to define for all $x : A$ and $y : B$:

$$f(x,y) :\equiv t$$

with $t : P(x,y)$.

Once again notations conflict with the (non-dependent) product, but this is okay because $A \times B$ behaves as $(x : A) \times \tilde{B}(x)$ where $\tilde{B}$ is the constant family in $A \to \mathcal{U}$ defined by $\tilde{B}(x) :\equiv B$.

For example we define:

$$p_1 : (x : A) \times B(x) \to A$$

$$p_2 : (z : (x : A) \times B(x)) \to B(p_1(z))$$

by $p_1(x,y) :\equiv x$ and $p_2(x,y) :\equiv y$.

3 First results about identity types

In this section we present some easy consequences of path induction, in order to familiarize the reader with this rather exotic way of reasoning. This will also make clear that the ∞-groupoid structure from the interpretation is (to some extent) already present in type theory.

3.1 Deriving some ∞-groupoid rules from path induction

In this section we show that path induction allows us to prove some rules of ∞-groupoids for types. First, we define composition of paths.

Lemma 1. *Let $A : \mathcal{U}$ be a type, and assume given $x, y, z : A$, with $p : x =_A y$ and $q : y =_A z$. Then, there is a path $p \cdot q : x =_A z$.*

Proof. By path induction we may assume that p and q are refl_x. But in this case we define:

$$\mathrm{refl}_x \cdot \mathrm{refl}_x :\equiv \mathrm{refl}_x$$

□

The path $p \cdot q$ is called the composition of p and q. Now we show that reflexivity is neutral for composition.

Lemma 2. *Let $A : \mathcal{U}$ be a type, assume given $x, y : A$ and $p : x =_A y$. Then we have:*

$$\mathrm{refl}_x \cdot p =_{x =_A y} p$$

and

$$p \cdot \mathrm{refl}_y =_{x =_A y} p$$

Proof. We only prove the first equality. By path induction we assume that p is refl_x, then we need to prove:

$$\mathrm{refl}_x \cdot \mathrm{refl}_x =_{x =_A x} \mathrm{refl}_x$$

But this is true by $\mathrm{refl}_{\mathrm{refl}_x}$. □

We show that composition of paths is associative.

Lemma 3. *Assume given $A : \mathcal{U}$ with $a, b, c, d : A$ and $p : a =_A b$, $q : b =_A c$, $r : c =_A d$. We can construct a path:*

$$(p \cdot q) \cdot r =_{a =_A d} p \cdot (q \cdot r)$$

Proof. By path induction we can assume that p, q and r are refl_a. In this case we need to build an inhabitant of:

$$(\mathrm{refl}_a \cdot \mathrm{refl}_a) \cdot \mathrm{refl}_a =_{a =_A a} \mathrm{refl}_a \cdot (\mathrm{refl}_a \cdot \mathrm{refl}_a)$$

But both sides are defined as refl_a, so $\mathrm{refl}_{\mathrm{refl}_a}$ is such an inhabitant. □

We construct the inverse of paths.

Lemma 4. *Let $A : \mathcal{U}$ be a type, and assume given $x, y : A$ with $p : x =_A y$. Then there is a path $p^{-1} : y =_A x$.*

Proof. By path induction we may assume that p is refl_x. But then we define:

$$(\mathrm{refl}_x)^{-1} :\equiv \mathrm{refl}_x$$

□

We show that inverses behave as expected.

Lemma 5. *Let $A : \mathcal{U}$ be a type, and assume given $x, y : A$ with $p : x =_A y$. Then there is a path in*

$$p \cdot p^{-1} =_{x=_A x} \mathrm{refl}_x$$

and

$$p^{-1} \cdot p =_{y=_A y} \mathrm{refl}_y$$

Proof. We just prove the first equality. By path induction we may assume that p is refl_x. Then we need to build an inhabitant of:

$$\mathrm{refl}_x \cdot (\mathrm{refl}_x)^{-1} =_{x=_A x} \mathrm{refl}_x$$

but the left-hand side is defined as refl_x, so $\mathrm{refl}_{\mathrm{refl}_x}$ is such an inhabitant. □

The general strategy for deriving rules of ∞-groupoids from path induction is quite clear from these examples: first we assume that all paths in sight are reflexivities, and then we just output a suitable reflexivity.

Remark 9. *In fact it has been proven that path induction implies that a type together with its iterated identity types has a structure of ∞-groupoid [20]. It is surprising that something as simple as path induction imply the rather complicated structure of ∞-groupoid.*

3.2 Functions as morphisms of ∞-groupoids

In this section we show that a map between types obeys some rules of morphisms between ∞-groupoids. We prove this as in the previous section: we do path induction on all paths in sight, and then output a suitable reflexivity. First we show that a map acts on paths.

Lemma 6. *Let $A, B : \mathcal{U}$ be types with $f : A \to B$. Assume given $x, y : A$ with $p : x =_A y$, then we have a term:*

$$\mathrm{ap}_f(p) : f(x) =_B f(y)$$

Proof. By path induction we assume that p is refl_x, but then we just define:

$$\mathrm{ap}_f(\mathrm{refl}_x) :\equiv \mathrm{refl}_{f(x)}$$

□

We show that this action commutes with composition of paths.

Lemma 7. *Assume given $A, B : \mathcal{U}$ and $f : A \to B$. If we have $x, y, z : A$ with $p : x =_A y$ and $q : y =_A z$, then:*

$$\mathrm{ap}_f(p \cdot q) =_{f(x)=_B f(z)} \mathrm{ap}_f(p) \cdot \mathrm{ap}_f(q)$$

Proof. By path induction we assume that both p and q are refl_x. Then we need to prove:

$$\mathrm{ap}_f(\mathrm{refl}_x \cdot \mathrm{refl}_x) =_{f(x)=_B f(x)} \mathrm{ap}_f(\mathrm{refl}_x) \cdot \mathrm{ap}_f(\mathrm{refl}_x)$$

But both sides are defined as $\mathrm{refl}_{f(x)}$, so we can conclude using $\mathrm{refl}_{\mathrm{refl}_{f(x)}}$ □

We show that the action of a morphism on paths commutes with inverses.

Lemma 8. *Assume given $A, B : \mathcal{U}$ and $f : A \to B$. If we have $x, y : A$ with $p : x =_A y$, then:*

$$\mathrm{ap}_f(p^{-1}) =_{f(y)=_B f(x)} (\mathrm{ap}_f(p))^{-1}$$

Proof. By path induction we assume that p is refl_x. Then we need to prove:

$$\mathrm{ap}_f(\mathrm{refl}_x^{-1}) =_{f(x)=_B f(x)} (\mathrm{ap}_f(\mathrm{refl}_x))^{-1}$$

But both sides are defined as $\mathrm{refl}_{f(x)}$, so $\mathrm{refl}_{\mathrm{refl}_{f(x)}}$ inhabits the desired type. □

3.3 Type families as families of ∞-groupoids

In this section we show that type families behave as families of homotopy types. First we define transport.

Lemma 9. *Let $A : \mathcal{U}$ be a type with $B : A \to \mathcal{U}$. Assume given $x, y : A$ and $p : x =_A y$, then we have a function:*

$$\mathbf{tr}_p^B : B(x) \to B(y)$$

Proof. By path induction it is enough to define:

$$\mathbf{tr}^B_{\mathrm{refl}_x}(b) :\equiv b$$

for $b : B(x)$. □

Next we show that transport interacts well with composition of paths.

Lemma 10. *Let $A : \mathcal{U}$ be a type with $B : A \to \mathcal{U}$. Assume given $x, y, z : A$ and $p : x =_A y$ with $q : y =_A z$. Then for all $b : B(x)$ we have:*

$$\mathbf{tr}_q^B \mathbf{tr}_p^B(b) =_{B(z)} \mathbf{tr}^B_{p \cdot q}(b)$$

Proof. By path induction we assume that p and q are refl_x. Then we need to prove:

$$\mathbf{tr}^B_{\mathrm{refl}_x} \mathbf{tr}^B_{\mathrm{refl}_x}(b) =_{B(x)} \mathbf{tr}^B_{\mathrm{refl}_x \cdot \mathrm{refl}_x}(b)$$

But both sides are defined as b, so refl_b has the desired type. □

Now we show that dependent functions interact well with family of paths.

Lemma 11. *Assume given $A : \mathcal{U}$ with $B : A \to \mathcal{U}$ and $f : (x : A) \to B(x)$. Then for $x, y : A$ and $p : x =_A y$ we have a term:*

$$\mathrm{apd}_f(p) : \mathbf{tr}_p^B(f(x)) =_{B(y)} f(y)$$

Proof. By path induction we assume that p is refl_x. Then we need to give an inhabitant of:

$$\mathbf{tr}^B_{\mathrm{refl}_x}(f(x)) =_{B(x)} f(x)$$

but since the left hand side is defined as $f(x)$, so we see that it is enough to define:

$$\mathrm{apd}_f(\mathrm{refl}_x) :\equiv \mathrm{refl}_{f(x)}$$

□

This long string of results shows path induction in action. From now on we will often use these results, and other ones obtained the same way, without any reference.

3.4 Identities in (dependent) function types

Definition 3. *Assume given $A, B : \mathcal{U}$ and $f, g : A \to B$. Then we define:*

$$f \sim g :\equiv (x : A) \to f(x) =_B g(x)$$

An inhabitant of $f \sim g$ is called a homotopy from f to g.

Remark 10. *As a map of ∞-groupoids, an inhabitant of $(x : A) \to f(x) =_B g(x)$ gives a lot more than a path from $f(x)$ to $g(x)$ for each point x in A, indeed it gives a continuous family of such paths.*

This can be extended straightforwardly to dependent functions:

Definition 4. *Assume given $A : \mathcal{U}$ with $B : A \to \mathcal{U}$ and $f, g : (x : A) \to B(x)$. Then we define:*

$$f \sim g :\equiv (x : A) \to f(x) =_{B(x)} g(x)$$

We see that the non-dependent case is a special case of the dependent one.

Lemma 12. *Assume given $A : \mathcal{U}$ with $B : A \to \mathcal{U}$ and $f, g : (x : A) \to B(x)$. Then the canonical function in:*

$$f = g \to f \sim g$$

is an equivalence.

The proof of this lemma uses the soon to be introduced *univalence axiom* (together with the notion of equivalence). It is a bit tricky, so we admit it. It can be found in section 4.9 of [19].

3.5 The univalence axiom: identity types in the universe

Now we define equivalences between types, which are interpreted as homotopy equivalences (or equivalences of ∞-groupoids).

Definition 5. *Let A, B be types in $\mathcal{U}$. A proof that $f : A \to B$ is an equivalence consists of:*

- *A map $g_1 : B \to A$ with a homotopy in $g_1 \circ f \sim \mathbf{id}_A$.*
- *A map $g_2 : B \to A$ with a homotopy in $f \circ g_2 \sim \mathbf{id}_B$.*

Formally, we define $\mathrm{isEquiv} : (A \to B) \to \mathcal{U}$ by:

$$\mathrm{isEquiv}(f) :\equiv (g_1 : B \to A) \times (g_2 : B \to A) \times (g_1 \circ f \sim \mathbf{id}_A) \times (f \circ g_2 \sim \mathbf{id}_B)$$

The use of g_1 and g_2 might surprise you, but it is necessary to guarantee the next lemma.

Lemma 13. *For all $\epsilon_1, \epsilon_2 : \mathrm{isEquiv}(f)$, we have:*

$$\epsilon_1 =_{\mathrm{isEquiv}(f)} \epsilon_2$$

This is desirable, because we want to identify an equivalence with its underlying map. In order to do that we need all proofs of equivalence to be equal. When proving that a map f is an equivalence, we will always choose $g_1 \equiv g_2$.

Remark 11. *We develop on this matter. In fact if we define:*

$$\mathrm{isQuasiEquiv}(f) :\equiv (g : B \to A) \times \big((x : A) \to g(f(x)) =_A x\big) \times \big((y : B) \to f(g(y)) =_B y\big)$$

One can prove that for all equivalence, we have $(y : B) \to g_1(y) =_A g_2(y)$, so that for all $f : A \to B$, we have

$$\mathrm{isEquiv}(f) \leftrightarrow \mathrm{isQuasiequiv}(f)$$

But the previous lemma does not hold for isQuasiEquiv*, so it is not equivalent to* isEquiv.

We introduce some notations.

Definition 6. *For A, B types in $\mathcal{U}$, we define the type of equivalences between A and B as:*

$$A \simeq B :\equiv (f : A \to B) \times \mathrm{isEquiv}(f)$$

Now the very important *univalence axiom.*

Assumption 5. *For all $A, B : \mathcal{U}$, we define* **ua** $: (A =_{\mathcal{U}} B) \to (A \simeq B)$ *by path induction:*

$$\mathbf{ua}(\mathrm{refl}_A) :\equiv \mathbf{id}_A$$

The univalence axiom says that this map is an equivalence.

The univalence axiom can be summarized (somewhat imprecisely) as:

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

Remark 12. *From a foundational point of view this is very interesting, especially for a category theorist. Indeed this can be seen as a formalisation of the principle that isomorphic objects are the same: in HoTT we cannot distinguish such objects.*

Remark 13. *For a homotopy theorist the univalence axiom is both a blessing and a curse:*

- *It is a blessing because whatever we define is automatically a homotopy invariant.*
- *It is a curse because we cannot use intermediate constructions not invariant under homotopy equivalences.*

3.6 Identities in (dependent) product types

The product of types is an example of inductive type. We want to perform an analysis of its identity types, i.e. find a type equivalent to $(x,y) =_{A\times B} (x',y')$ for $x,x' : A$ and $y,y' : B$. In order to do that we introduce the *encode-decode method*, an essential tool of homotopy type theory.

Proposition 1. *Assume given $x,x' : A$ and $y,y' : B$. Then we have:*

$$(x,y) =_{A\times B} (x',y') \simeq (x =_A x') \times (y =_B y')$$

If we try to build directly a map out of $(x,y) =_{A\times B} (x',y')$, we cannot proceed by path induction because (x,y) and (x',y') are not variables. In fact this problem will be encountered whenever we try to analyse the identity types of an inductive type. We present a solution called the *encode-decode method*. The key idea is to generalize our property to $z =_{A\times B} z'$.

Proof. We define $\mathrm{Eq}_{A\times B} : A\times B \to A\times B \to \mathscr{U}$ inductively by:

$$\mathrm{Eq}_{A\times B}((x,y),(x',y')) :\equiv (x =_A x') \times (y =_B y')$$

It is clear that proving:

$$(z,z' : A\times B) \to z =_{A\times B} z' \simeq \mathrm{Eq}_{A\times B}(z,z')$$

will imply the desired property.

First we define $\mathbf{encode} : z =_{A\times B} z' \to \mathrm{Eq}_{A\times B}(z,z')$. By path induction it is enough to define $\mathbf{encode}(\mathrm{refl}_z) : \mathrm{Eq}_{A\times B}(z,z)$ for all $z : A\times B$, but assuming by induction that z is (x,y), it is enough to define:

$$\mathbf{encode}(\mathrm{refl}_{(x,y)}) :\equiv (\mathrm{refl}_x, \mathrm{refl}_y)$$

Now we define **decode** : $\mathrm{Eq}_{A\times B}(z,z') \to z =_{A\times B} z'$. By induction we assume z is (x,y) and z' is (x',y'). Then we need to give:

$$\mathbf{decode} : (x =_A x') \times (y =_B y') \to (x,y) =_{A\times B} (x',y')$$

By induction on the product and two path inductions, it is enough to define:

$$\mathbf{decode}(\mathrm{refl}_x, \mathrm{refl}_y) :\equiv \mathrm{refl}_{(x,y)}$$

We can prove that for all $z, z' : A \times B$ we have:

- For all $q : \mathrm{Eq}_{A\times B}(z,z')$
$$\mathbf{encode}(\mathbf{decode}(q)) = q$$
indeed by induction on z, z' followed by induction on q, it is enough to prove:
$$\mathbf{encode}(\mathbf{decode}(\mathrm{refl}_x, \mathrm{refl}_y)) = (\mathrm{refl}_x, \mathrm{refl}_y)$$
but this is true by reflexivity.

- For all $p : z =_{A\times B} z'$ we have:
$$\mathbf{decode}(\mathbf{encode}(p)) = p$$
indeed by induction on p followed by induction on z, it is enough to prove:
$$\mathbf{decode}(\mathbf{encode}(\mathrm{refl}_{(x,y)})) = \mathrm{refl}_{(x,y)}$$
but the constant path solves this as well.

Now we can conclude. □

I was very chatty in this proof. I hope it will help you replicate this encode-decode method for other inductive types (see TD1). Later on we will see how it can be made shorter using the appropriate lemma.

Now we want to perform a similar analysis of the identity types in dependent products.

Lemma 14. *Assume given $A : \mathcal{U}$ with $B : A \to \mathcal{U}$. For $x, x' : A$ and $y : B(x)$ and $y' : B(x')$, we have:*

$$(x,y) =_{(a:A)\times B(a)} (x',y') \;\simeq\; (p : x =_A x') \times (\mathbf{tr}^B_p(y) =_{B(x')} y')$$

Proof. The proof is the same as for the non-dependent case. □

Note that we need to use the transport $\mathbf{tr}^B_p$ because $y : B(x)$ whereas $y' : B(x')$, so they are not in the same type.

Exercise 2. *Convince yourself that the proof for non-dependent products works in the dependent case.*

3.7 Using univalence

We give an example of use of univalence, and comment a bit on it. First we define the type of maps to B for $B : \mathcal{U}$.

Definition 7. *For $B : \mathcal{U}$, we define:*

$$\mathcal{U}_{/B} :\equiv (X : \mathcal{U}) \times (X \to B)$$

For $f : A \to B$, we write $f : \mathcal{U}_{/B}$ as a shorthand for (A, f). As usual, we want to characterize identity types in $\mathcal{U}_{/B}$.

Lemma 15. *Assume given $A, A', B : \mathcal{U}$ with $f : A \to B$ and $g : A' \to B$ then:*

$$f =_{\mathcal{U}_{/B}} g$$

is equivalent to the type:

$$(\epsilon : A \simeq A') \times (f \sim g \circ \epsilon)$$

Proof. We need to characterize:

$$(A, f) =_{(X:\mathcal{U})\times(X\to B)} (A', g)$$

But by the characterization of identity types in a dependent product, it is equivalent to:

$$(\epsilon : A =_{\mathcal{U}} A') \times (\mathbf{tr}_{\epsilon}^{\lambda Z.Z\to B}(f) =_{A'\to B} g)$$

- Now we prove that for any $\epsilon : A =_{\mathcal{U}} A'$ and $f : A \to B$, $g : A' \to B$ we have:

  $$\left(\mathbf{tr}_{\epsilon}^{\lambda Z.Z\to B}(f) =_{A'\to B} g\right) \simeq \left(f =_{A\to B} g \circ \mathbf{ua}(\epsilon)\right)$$

  Where **ua** is the map from $A =_{\mathcal{U}} B$ to $A \simeq B$ is assumed to be an equivalence by the univalence axiom. By path induction on ϵ one just needs to show that:

  $$\left(\mathbf{tr}_{\mathrm{refl}_A}^{\lambda Z.Z\to B}(f) =_{A\to B} g\right) \simeq \left(f =_{A\to B} g \circ \mathbf{ua}(\mathrm{refl}_A)\right)$$

  but by definition this just means:

  $$\left(f =_{A\to B} g\right) \simeq \left(f =_{A\to B} g\right)$$

  which is obvious.

From that we know that $f =_{\mathcal{U}_{/B}} g$ is equivalent to:

$$(\epsilon : A =_{\mathcal{U}} A') \times (f =_{A\to B} g \circ \mathbf{ua}(\epsilon))$$

But using univalence (i.e. the fact that **ua** is an equivalence) we can show (omitted) that this is equivalent to:

$$(\epsilon : A \simeq A') \times (f =_{A\to B} g \circ \epsilon)$$

which is equivalent to what we want by function extensionality.

□

You should convince yourself that this result was to be expected: what else could reasonably be called an equivalence between maps to B ?

Remark 14. *This illustrates a general situation: assume we define a notion of "types with extra-structure" (e.g. maps to B).*

- *In traditional mathematics one would need to define by hand a notion of "equivalences" or "isomorphisms" of such types with extra-structure, and then prove that "equivalent" objects share the same properties.*
- *In homotopy type theory we just use plain equality as the correct notion of "equivalence", so that equivalent objects automatically share the same properties. Then univalence (and a bit of work) shows that this is the same as the traditional notion mathematicians came up with.*

3.8 A first theorem: Grothendieck correspondence for types

We give a very important auxiliary definition.

Definition 8. *Assume given a map $f : X \to Y$. We define $\mathbf{fib}_f : Y \to \mathcal{U}$ by:*

$$\mathbf{fib}_f(y) :\equiv (x : X) \times f(x) =_X y$$

Note that there is a canonical map $p_1 : \mathbf{fib}_f(y) \to X$.

Remark 15. *Here the set-theoretic perspective is that $\mathbf{fib}_f(y)$ is what is usually denoted $f^{-1}(y)$. We will see that this analogy fails to explain some phenomena. A far better analogy is with the* homotopy fibers *in traditional algebraic topology.*

We need an auxiliary lemma.

Lemma 16. *Assume given $Y : \mathcal{U}$ with $P : Y \to \mathcal{U}$ and $b : Y$. Then we have*

$$(z : (y : Y) \times P(y)) \times (p_1(z) =_Y b) \simeq P(b)$$

Proof. We just construct the equivalence:

- We define $\phi : P(b) \to (z : (y : Y) \times P(y)) \times (p_1(z) =_Y b)$ by:

  $$\phi(r) :\equiv ((b, r), \mathrm{refl}_b)$$

- We need to define:

  $$\psi : (z : (y : Y) \times P(y)) \times (p_1(z) =_Y b) \to P(b)$$

  Inductively on z it is enough to define for $y : Y$ and $r : P(y)$ with $q : y =_Y b$ a term:

  $$\psi((y, r), q) : P(b)$$

  But by based path induction on q (this is possible because y is a variable), it is enough to define:

  $$\psi((b, r), \mathrm{refl}_b) :\equiv r$$

- We have that $\psi(\phi(r)) =_{P(b)} r$ by definition for $r : P(b)$.
- To show that $\phi(\psi(z)) = z$ we proceed by induction on z, so we just need to prove:
$$\phi(\psi((y,r),q) = ((y,r),q)$$
for $y : Y$ with $r : P(y)$ and $q : y =_Y b$. But by based path induction on q, we just need to prove:
$$\phi(\psi((b,r),\mathrm{refl}_b)) = ((b,r),\mathrm{refl}_b)$$
which is true by definition.

□

Now we prove our first theorem.

Theorem 1. *For $Y : \mathcal{U}$, we have:*

$$(Y \to \mathcal{U}) \simeq \mathcal{U}_{/Y}$$

Proof. We define $\psi : (Y \to \mathcal{U}) \to \mathcal{U}_{/Y}$ by:

$$\psi(P) :\equiv ((y : Y) \times P(y), p_1)$$

for $P : Y \to \mathcal{U}$. The other way we define $\phi : \mathcal{U}_{/Y} \to (Y \to \mathcal{U})$ by induction as:

$$\psi((Y,f)) :\equiv \mathbf{fib}_f$$

Now we need to check that both constructions are inverse to each other.

- For $P : Y \to \mathcal{U}$ we need to prove $\phi(\psi(P)) =_{Y\to\mathcal{U}} P$, i.e. with:
$$p_1 : (y : Y) \times P(y) \to Y$$
we need:
$$\mathbf{fib}_{p_1} =_{Y\to\mathcal{U}} P$$
But this is equivalent by function extensionality to proving for all $b : Y$ that:
$$\mathbf{fib}_{p_1}(b) =_{\mathcal{U}} P(b)$$
But this is implied by the previous lemma together with univalence.
- In the other direction assume given $f : X \to Y$, we want to show that for:
$$p_1 : (y : Y) \times \mathbf{fib}_f(y) \to Y$$
we have:
$$p_1 =_{\mathcal{U}_{/Y}} f$$
By the characterisation of identity types in $\mathcal{U}_{/Y}$, it is enough to give an equivalence:
$$\epsilon : X \to (y : Y) \times \mathbf{fib}_f(y)$$

such that for all $x : X$, we have $p_1(\epsilon(x)) =_Y f(x)$.

We define ϵ by:

$$\epsilon(x) :\equiv (f(x), x, \mathrm{refl}_{f(x)})$$

It is obvious that ϵ obeys the desired equality. Now we check that ϵ is an equivalence. To do that we define:

$$\alpha : (y : Y) \times \mathbf{fib}_f(y) \to X$$

inductively by:

$$\alpha(y, x, r) :\equiv x$$

where $y : Y$, $x : X$ and $r : f(x) =_Y y$.

Then $\alpha(\epsilon(x))$ is by definition equal to x. We want to prove $\epsilon(\alpha(z)) = z$, so by induction we just need to prove that for $y : Y$, $x : X$ and $r : f(x) =_Y y$, we have that:

$$\epsilon(\alpha(y, x, r)) = (y, x, r)$$

i.e.

$$(f(x), x, \mathrm{refl}_{f(x)}) = (y, x, r)$$

But we can assume that r is $\mathrm{refl}_{f(x)}$ (so that y is $f(x)$) by based path induction (since r has one variable y as its endpoint). Then it is easy to conclude.

□

Remark 16. *A similar result is true in set theory. Roughly it says that for Y a set, giving a family of sets $(X_y)_{y \in Y}$ is "equivalent" (in some not-so-clear sense at this point) to giving a function to Y. In this case we construct $\psi((X_y)_{y \in Y})$ as $\coprod_y X_y \to Y$ (where $\coprod$ is the so-called "disjoint union", because it is in bijection with the union $\cup_y X_y$ when the X_y are disjoints) and $\phi(f)$ is $(f^{-1}(y))_{y \in Y}$.*

We see that type theory has two advantages over set theory in this case:

- *Knowing in which sense families of sets are "equivalent" to functions require to know a bit of category theory (technically we use equivalences of category). In homotopy type theory families correspond to maps in a very canonical way: by univalence we have $(Y \to \mathcal{U}) = \mathcal{U}_{/Y}$. This means that (at least in this case) homotopy type theory has some foundational value: it gives a more natural meaning to this "families equivalent to maps" idea.*

- *On the other hand, using the homotopical interpretation of type theory we deduce from our theorem that for Y an ∞-groupoid, the ∞-groupoid of morphisms (of ∞-groupoid) to Y is equivalent to the ∞-groupoid of morphisms (of ∞-groupoid) from Y to the ∞-groupoid of all ∞-groupoids. Proving (and even stating...) this theorem in traditional mathematics requires a lot of serious work. But our proof does not require a lot more work than for sets.*

4 Inductive types and the encode-decode method

In this section we give examples of inductive types. These types are defined by a list of constructors allowing us to build inhabitants of them, and an elimination stating that they are the simplest types with these constructors. We do not attempt to give a general definition of inductive types. One can be found in [11].

4.1 The unit type

The unit type $\mathbf{1}$ corresponds to a singleton $\{*\}$ in set theory, and the true proposition $\top$ in logic.

Inductive Definition 4. *We define the unit type.*

- *(Formation) There is a type* $\mathbf{1} : \mathcal{U}$.
- *(Introduction) There is a term* $* : \mathbf{1}$.
- *(Elimination) Assume given* $P : \mathbf{1} \to \mathcal{U}$. *In order to define:*

$$f : (x : \mathbf{1}) \to P(x)$$

it is enough to define:

$$f(*) :\equiv t$$

with $t : P(*)$.

Lemma 17. *For any* $x : \mathbf{1}$, *we have* $x = *$

Proof. By induction we can assume that x is $*$. Then the property is obvious. □

Lemma 18. *Assume given* $x, y : \mathbf{1}$, *then we have:*

$$(x = y) \simeq \mathbf{1}$$

Proof. See TD1. □

4.2 The empty type

We look at a somewhat degenerate case. The empty type $\mathbf{0}$ is defined by having no constructor, meaning there is no way to build an element in it. This makes it akin to the empty set $\emptyset$ from set theory, and the false proposition $\bot$ from logic.

Inductive Definition 5. *We define the empty set.*

- *(Formation) There is a type* $\mathbf{0} : \mathcal{U}$.
- *(Introduction) There is no introduction rule.*

- *(Elimination) Assume given $P : \mathbf{0} \to \mathcal{U}$, in order to define a map:*

$$f : (x : \mathbf{0}) \to P(x)$$

we do not need to give anything!

In practice the elimination principle means that if we assume $x : \mathbf{0}$, we can immediately conclude. We do not need any subtlety to study identity types in $\mathbf{0}$, in fact we can prove anything about them!

The type $\mathbf{0}$ is used to define the negation of a type $\neg A :\equiv A \to \mathbf{0}$. If A is seen as a set then an inhabitant of $\neg A$ gives the mean to build an inhabitant of any type from an inhabitant of A, intuitively meaning that A is empty, if A is seen as a proposition then an inhabitant of $\neg A$ is a refutation of A.

4.3 The type of booleans

The type of booleans $\mathbf{2}$ correspond to a set with two elements. Traditionally these elements are called **true** and **false** by computer scientists, explaining the name of booleans. We will call them 0 and 1, hoping that no confusion with the natural numbers occurs.

Inductive Definition 6. *We define the type of booleans.*

- *(Formation) There is a type $\mathbf{2} : \mathcal{U}$.*
- *(Introduction) We have elements $0, 1 : \mathbf{2}$.*
- *(Elimination) Assume given $P : A \to \mathcal{U}$, in order to define:*

$$f : (x : \mathbf{2}) \to P(x)$$

it is enough to define:

$$f(0) :\equiv s$$
$$f(1) :\equiv t$$

with $s : P(0)$ and $t : P(1)$.

Its identity types are as expected.

Lemma 19. *We have that:*

$$(0 =_{\mathbf{2}} 0) \simeq \mathbf{1}$$
$$(0 =_{\mathbf{2}} 1) \simeq \mathbf{0}$$
$$(1 =_{\mathbf{2}} 0) \simeq \mathbf{0}$$
$$(1 =_{\mathbf{2}} 1) \simeq \mathbf{1}$$

Proof. See TD1. □

4.4 Sum types

We can also define types depending on other types. For example we define the sum type $A + B$ for $A, B : \mathcal{U}$, which corresponds to the disjoint union of A and B from a set theoretic perspective.

Inductive Definition 7. *We define sum types.*

- *(Formation) Assume given $A, B : \mathcal{U}$, then there is a type $A + B : \mathcal{U}$.*
- *(Introduction) For any $x : A$ we have an element:*

  $$\mathbf{inc}_l(x)$$

  Moreover for all $y : B$ we have an element:

  $$\mathbf{inc}_r(y)$$

- *(Elimination) Assume given $P : A + B \to \mathcal{U}$, in order to define:*

  $$f : (x : A + N) \to P(x)$$

  it is enough to define:

  $$f(\mathbf{inc}_l(x)) :\equiv s$$

  with $s : P(\mathbf{inc}_l(x))$ for $x : A$ and:

  $$f(\mathbf{inc}_l(y)) :\equiv t$$

  with $t : P(\mathbf{inc}_r(y))$ for $y : B$.

Lemma 20. *Assume given $A, B : \mathcal{U}$ with $a, a' : A$ and $b, b' : B$. Then:*

$$\mathbf{inc}_l(a) = \mathbf{inc}_l(a') \simeq a =_A a'$$

$$\mathbf{inc}_l(a) = \mathbf{inc}_r(b') \simeq \mathbf{0}$$

$$\mathbf{inc}_r(b) = \mathbf{inc}_l(a') \simeq \mathbf{0}$$

$$\mathbf{inc}_r(b), \mathbf{inc}_l(b') \simeq b =_B b'$$

Proof. See TD1. □

4.5 The natural numbers

We introduce our first truly recursive inductive type.

Inductive Definition 8. *We define the type of natural numbers.*

- *(Formation) There is a type $\mathbb{N} : \mathcal{U}$.*
- *(Introduction) We have elements $0 : \mathbb{N}$, and we have a function:*

  $$\mathbf{s} : \mathbb{N} \to \mathbb{N}$$

- *(Elimination) Assume given $P : \mathbb{N} \to \mathcal{U}$, in order to define:*

$$(x : \mathbb{N}) \to P(x)$$

it is enough to define:

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

$$f(\mathbf{s}(n)) :\equiv t$$

with $s : P(0)$ and $t : P(\mathbf{s}(n))$, which can be build using $f(n)$.

The idea is that $\mathbf{s}$ is the successor function (which sends n to $n+1$). This type is called recursive because the constructor $\mathbf{s}$ use an element in $\mathbb{N}$ in order to build a new one. You should convince yourself that the elimination principle for $\mathbb{N}$ is just the usual induction on natural numbers.

Lemma 21. *Assume given $m, n : \mathbb{N}$, then:*

$$0 =_{\mathbb{N}} 0 \simeq \mathbf{1}$$

$$\mathbf{s}(m) =_{\mathbb{N}} 0 \simeq \mathbf{0}$$

$$0 =_{\mathbb{N}} \mathbf{s}(n) \simeq \mathbf{0}$$

$$\mathbf{s}(m) =_{\mathbb{N}} \mathbf{s}(n) \simeq m =_{\mathbb{N}} n$$

Proof. See TD1. □

4.6 The integers

We introduce $\mathbb{Z}$.

Inductive Definition 9. *We define non-zero natural number.*

- *There is a type $\mathbb{N}^*$.*
- *There is an inhabitant $1 : \mathbb{N}^*$, and for any $m : \mathbb{N}^*$ there is an inhabitant:*

$$\mathbf{s}(m) : \mathbb{N}^*$$

- *The induction principle is the same as for natural numbers.*

Note that $\mathbb{N}^*$ is equivalent to $\mathbb{N}$, we just gave different name to its inhabitants.

Inductive Definition 10. *We define integers.*

- *There is a type $\mathbb{Z}$.*
- *We have an inhabitant $0 : \mathbb{Z}$. For any $m : \mathbb{N}^*$, we have two inhabitants:*

$$+m : \mathbb{Z}$$

$$-m : \mathbb{Z}$$

- *Assume we want to define:*
$$f : (x : \mathbb{Z}) \to P(x)$$
then it is enough to define:
$$f(0) :\equiv s_0$$
$$f(+m) :\equiv s_+$$
$$f(-m) :\equiv s_-$$
For s_0, s_+ and s_- of the appropriate type.

Note that integers behave as the usual set-theoretic integers, so that we will use addition, multiplication, etc without proof.

Exercise 3. *Analyse identity types in $\mathbb{Z}$ using the encode-decode method.*

5 Higher inductive types

In this section we give an introduction to higher inductive types. They are inductive types with constructors in their iterated identity types.

5.1 The circle

We give our first example of higher inductive type.

Definition 9. *We have:*

- *(Formation) There is a type S^1.*
- *(Introduction) There is a point:*
$$\mathbf{base} : S^1$$
Moreover there is a path:
$$\mathbf{loop} : \mathbf{base} =_{S^1} \mathbf{base}$$
- *(Elimination) In order to define $f : (x : S^1) \to P(x)$ it is enough to define:*
$$f(\mathbf{base}) :\equiv t$$
$$\mathrm{apd}_f(\mathbf{loop}) :\equiv \epsilon$$
For $t : P(\mathbf{base})$ and
$$\epsilon : \mathbf{tr}^P_{\mathbf{loop}}(t) =_{P(\mathbf{base})} t$$

This is a type freely generated by a point **base** and an identity proof **loop**. As a space this correspond to a point b with a path from this point to itself added, i.e. a circle. So we call S^1 the circle.

5.2 Identity types in the circle

As usual we want to characterize the identity types in S^1. One can guess that $\mathbf{base} =_{S^1} \mathbf{base}$ will have one point corresponding to $\mathrm{refl}_{\mathbf{base}}$, plus one point corresponding to $\mathbf{loop}$. But as S^1 is a type there should also be a point for $\mathbf{loop} \cdot \mathbf{loop}$, which should be different from the two previous one (as S^1 is freely generated). There should be a point for $\mathbf{loop}^{-1}$ as well, and so on.

From these hints we guess that $\mathbf{base} =_{S^1} \mathbf{base}$ is equivalent to $\mathbb{Z}$, via the map sending $n : \mathbb{Z}$ to $\mathbf{loop}^n$. When proving this we need to generalize this to $\mathbf{base} =_{S^1} x$, so that we can use path induction.

Definition 10. *We define* $\mathrm{Eq}_{S^1} : S^1 \to \mathcal{U}$ *inductively by:*

$$\mathrm{Eq}_{S^1}(\mathbf{base}) :\equiv \mathbb{Z}$$

$$\mathrm{ap}_{\mathrm{Eq}_{S^1}}(\mathbf{loop}) :\equiv \mathbf{ua}(\lambda(x : \mathbb{Z}).\, x + 1)$$

Now we want to show that $\mathrm{Eq}_{S^1}(x)$ is equivalent to $\mathbf{base} =_{S^1} x$. We use auxiliary lemmas.

Lemma 22. *For all* $m : \mathbb{Z}$*, we have that* $\mathbf{tr}^{\mathrm{Eq}_{S^1}}_{\mathbf{loop}}(m) = m + 1$.

Proof. We prove that for any $B : A \to \mathcal{U}$ and $p : x =_A y$ we have $\mathbf{tr}^B_p = \mathbf{ua}(\mathrm{ap}_B(p))$ by path induction. But then we know that by definition $\mathrm{ap}_{\mathrm{Eq}^{S^1}}(\mathbf{loop}) = \mathbf{ua}^{-1}(\lambda m.m+1)$, so we can conclude. □

Lemma 23. *For any* $x : S^1$ *we have functions:*

$$\mathbf{encode} : \mathbf{base} =_{S^1} x \to \mathrm{Eq}_{S^1} x$$

and:

$$\mathbf{decode} : \mathrm{Eq}_{S^1}(x) \to \mathbf{base} =_{S^1} x$$

Proof. We define **encode** by path induction, with:

$$\mathbf{encode}(\mathrm{refl}_{\mathbf{base}}) :\equiv 0$$

In order to define **decode**, we proceed by induction on $x : S^1$:

- If x is **base**, we need to provide a function in:

  $$\mathbb{Z} \to \mathbf{base} = \mathbf{base}$$

  For this we just use $\lambda(n : \mathbb{Z}).\mathbf{loop}^n$ defined by:

  $$\mathbf{loop}^0 :\equiv \mathrm{refl}_{\mathbf{base}}$$

  $$\mathbf{loop}^{+n} :\equiv \mathbf{loop} \cdot \ldots \cdot \mathbf{loop}$$

  with n occurences of **loop**, and similarly:

  $$\mathbf{loop}^{-n} :\equiv \mathbf{loop}^{-1} \cdot \ldots \cdot \mathbf{loop}^{-1}$$

- For the **loop** case, we need to prove that:

$$\mathbf{tr}_{\mathbf{loop}}^{\lambda x.\mathrm{Eq}_{S^1}(x)\to\mathbf{base}=x}(\mathbf{loop}^{-}) =_{\mathbb{Z}\to\mathbf{base}=\mathbf{base}} \mathbf{loop}^{-}$$

  We can see that this is equivalent to proving that for all $n:\mathbb{Z}$ we have:

$$\mathbf{tr}_{\mathbf{loop}}^{\lambda x.\mathbf{base}=x}(\mathbf{loop}^{\mathbf{tr}_{\mathbf{loop}^{-1}}^{\mathrm{Eq}_{S^1}}(n)}) = \mathbf{loop}^{n}$$

  because $\mathbf{tr}_{p}^{\lambda x.P(x)\to Q(x)}(f) = \mathbf{tr}_{p}^{Q} \circ f \circ \mathbf{tr}_{p^{-1}}^{P}$. Then we just need to prove:

$$\mathbf{tr}_{\mathbf{loop}}^{\lambda x.\mathbf{base}=x}(\mathbf{loop}^{n-1}) = \mathbf{loop}^{n}$$

  which is in turn equivalent to:

$$\mathbf{loop}^{n-1} \cdot \mathbf{loop} = \mathbf{loop}^{n}$$

  using the fact that $\mathbf{tr}_{q}^{\lambda x.a=_{A}x}(p) = p \cdot q$ for any type A.
  But this last statement is more or less the definition of **loop**$^{-}$.

□

Lemma 24. *For any $n:\mathbb{Z}$ we have:*

$$\mathbf{encode}(\mathbf{loop}^{n}) = n$$

Proof. We check by path induction on p that $\mathbf{encode}(p) = \mathbf{tr}_{p}^{\mathrm{Eq}_{S^1}}(0)$. Then we need to check that:

$$\mathbf{tr}_{\mathbf{loop}^{n}}^{\mathrm{Eq}_{S^1}}(0) = n$$

But we know that:

$$\mathbf{tr}_{\mathbf{loop}}^{\mathrm{Eq}_{S^1}}(m) = m+1$$

Using induction on $n:\mathbb{Z}$ we can conclude. □

Exercise 4. *Do the induction on $n:\mathbb{Z}$.*

Proposition 2. *For all $x:S^1$ we have:*

$$(\mathbf{base} =_{S^1} x) \;\simeq\; \mathrm{Eq}_{S^1}(x)$$

Proof. We use the encode-decode method as usual.

- We use:

$$\mathbf{encode}:(x:S^1)\to(\mathbf{base}=x)\to\mathrm{Eq}_{S^1}(x)$$

$$\mathbf{decode}:(x:S^1)\to\mathrm{Eq}_{S^1}(x)\to(\mathbf{base}=_{S^1}x)$$

  defined in the previous lemma.

- We prove that for all $x : S^1$ and $p : \mathbf{base} =_{S^1} x$, we have:

$$\mathbf{decode}(\mathbf{encode}(p))$$

by path induction on p.

- Finally we prove that for all $x : S^1$ we have:

$$(q : \mathrm{Eq}_{S^1}(x)) \to \mathbf{encode}(\mathbf{decode}(q)) = q$$

We proceed by induction on x.

If x is **base**, we need to prove that for all $n : \mathbb{Z}$:

$$\mathbf{encode}(\mathbf{loop}^n) =_{\mathbb{Z}} n$$

but this is the previous lemma.

If x is **loop** then we need to show that:

$$\mathbf{tr}_{\mathbf{loop}}(H) =_{(n:\mathbb{Z}) \to \psi(\mathbf{loop}^n) =_{\mathbb{Z}} n} H$$

where H is the previous proof of $(n : \mathbb{Z}) \to \mathbf{encode}(\mathbf{loop}^n) = n$. By function extensionality it is enough to show that for all $n : \mathbb{Z}$ we have:

$$\mathbf{tr}_{\mathbf{loop}}(H)(n) =_{\psi(\mathbf{loop}^n) =_{\mathbb{Z}} n} H(n)$$

But using an analysis of identity types in $\mathbb{Z}$ by the encode-decode method, we can prove that for any $p, q : m =_{\mathbb{Z}} n$ we have $p = q$.

□

Proposition 3. *We have:*

$$(\mathbf{base} =_{S^1} \mathbf{base}) \simeq \mathbb{Z}$$

Proof. This is an immediate consequence of our last result. □

5.3 The suspension

Now we will give an example of higher inductive type which will be useful.

Inductive Definition 11. *We define the suspension ΣX of a type $X : \mathcal{U}$ by:*

- *(Formation) For any $X : \mathcal{U}$, there is a type $\Sigma X : \mathcal{U}$.*
- *(Introduction) We have two points $\mathbf{N}, \mathbf{S} : \Sigma X$. For any $x : X$, we have a path:*

$$\mathbf{merid}_x : \mathbf{N} =_{\Sigma X} \mathbf{S}$$

- *(Elimination) In order to define $f : (x : \Sigma X) \to P(x)$, it is enough to define:*

$$f(\mathbf{N}) :\equiv s$$

$$f(\mathbf{S}) :\equiv t$$

$$\mathrm{apd}_f(\mathbf{merid}_x) :\equiv \epsilon_x$$

For $s : P(\mathbf{N})$ and $t : P(\mathbf{S})$, with:

$$\epsilon_x : \mathbf{tr}^P_{\mathbf{merid}_x}(s) =_{P(\mathbf{S})} t$$

The geometric picture is a bit complicated. For example we will see that $\Sigma\mathbf{2}$ is the circle and ΣS^1 is the sphere.

5.4 More higher inductive types

We did not attempt to define inductive types in general in these notes, a complete definition can be found in [11]. Higher inductive types are a variant with constructors not only for inhabitants of the type, but also of its iterated identity types. There is currently no generally accepted definition of higher inductive types, although some work in this direction has been done (see for example [8]). Just note that a definition of HITs would consist of:

- A rule determining which constructors are valid.
- An algorithm giving the elimination principle corresponding to a given list of constructors.

Here we give some examples of higher inductive types, but we do not state their dependent elimination principles.

The torus

Inductive Definition 12. *We define the torus T^2 by:*

- *(Formation) There is a type $T^2 : \mathcal{U}$.*
- *(Introduction) There is a point $b : T^2$, with:*

$$p, q : b =_{T^2} b$$

and moreover:

$$s : p \cdot q =_{b =_{T^2} b} q \cdot p$$

- *(Non-dependent Elimination) In order to define $f : T^2 \to A$, it is enough to define:*

$$f(b) :\equiv s$$

$$\mathrm{ap}_f(p) :\equiv \epsilon_1$$

$$\mathrm{ap}_f(q) :\equiv \epsilon_2$$

$$\mathrm{ap}^2_f(s) :\equiv h$$

for $s : A$, *with* $\epsilon_1, \epsilon_2 : s =_A s$ *and* $h : \epsilon_1 \cdot \epsilon_2 =_{s=_A s} \epsilon_2 \cdot \epsilon_1$.

We do not state the dependent elimination principle, although it is necessary in order for the torus to be well-behaved. It implies that:

$$(T^2 \to A) \simeq (x : A) \times (z_1, z_2 : x =_A x) \times (z_1 \cdot z_2 =_{x=_A x} z_2 \cdot z_1)$$

The geometric idea is simple: if we glue the opposite edges of a square together we obtain a torus. We can present a lot of spaces as HITs using this kind of decompositions.

Remark 17. *In algebraic topology such a decomposition of a space as some points, paths, surfaces between paths, etc is called a cellular decomposition. When we know a (say finite) cellular decomposition of a topological space, we can represent it as a HIT.*

The sphere

Now we present the sphere S^2.

Inductive Definition 13. *We define the sphere* S^2 *by:*

- *(Formation) There is a type* $S^2 : \mathcal{U}$.
- *(Introduction) There is a point* $\mathbf{base} : S^2$, *and there is a path:*

  $$h : \mathrm{refl}_{\mathbf{base}} =_{\mathbf{base}=_{S^2}\mathbf{base}} \mathrm{refl}_{\mathbf{base}}$$

  the picture you should have in mind is a disk whose border is constant equal to **base**.
- *(Non-dependent Elimination) In order to define* $f : S^2 \to A$, *it is enough to define:*

  $$f(\mathbf{base}) :\equiv t$$

  $$\mathrm{ap}^2_f(h) :\equiv t$$

  for $s : A$ *and* $t : \mathrm{refl}_s =_{s=_A s} \mathrm{refl}_s$.

The n-dimensional spheres can be defined in a similar way, for example S^3 has a point $\mathbf{base} : S^3$ and

$$h : \mathrm{refl}_{\mathrm{refl}_{\mathbf{base}}} =_{\mathrm{refl}_{\mathbf{base}}=_{S^3}\mathrm{refl}_{\mathbf{base}}} \mathrm{refl}_{\mathrm{refl}_{\mathbf{base}}}$$

Quotients

We define quotients as HITs.

Inductive Definition 14. *Assume given $A : \mathcal{U}$ and $\_ \sim \_ : A \to A \to \mathcal{U}$. We define the the quotient $A/_\sim$ of A by $\_ \sim \_$:*

- *(Formation) We have a type $A/_\sim$.*
- *(Introduction) For any $x : A$, we have:*

$$[x] : A/_\sim$$

For any $x, y : A$ and $p : x \sim y$, we have a path:

$$\mathrm{quo}(p) : [x] =_{A/_\sim} [y]$$

- *(Non-dependent Elimination) Assume given $B : \mathcal{U}$. In order to define*

$$f : A/_\sim \to B$$

it is enough to define:

$$f([x]) :\equiv s$$

$$\mathrm{ap}_f(\mathrm{quo}(p)) :\equiv e$$

for $s : B$ and $e : f([x]) =_B f([y])$ for $p : x \sim y$.

This is quite different from the usual quotients, as an example $\mathbf{1}$ quotiented by the relation

$$x \sim y :\equiv \mathbf{1}$$

is equivalent to S^1. They are called homotopy quotients in traditional homotopy theory. The usual quotients of sets can be defined in HoTT by taking the set-truncation of the homotopy quotients.

Remark 18. *It should be noted that quotients in set-theory are ill-behaved with respect to families. Indeed assume you have a family of sets $(X_a)_{a \in A}$ indexed by a set A, and and a relation $\_ \sim \_$ on A.Moreover assume that if $a \sim b$, we have a bijection between X_a and X_b. Then it is not possible to get a family of sets indexed over $A/_\sim$ in a satisfying way. In homotopy type theory this is possible, using quotient as HITs and univalence.*

This remark that quotients are better behaved in homotopy type theory than in set-theory supports univalent foundations. *They claim that univalence and HITs are more natural and desirable than the usual features of set-theory, even if you are not interested in homotopy types.*

And a lot more...

There are a lot of other HITs, not always with a geometric flavor. We give examples of things definable as HITs:

- Real numbers.
- Free groups (or more generally free algebraic structure).
- The cumulative hierarchy V.
- Type theory!
- ...

6 Truncation properties

In this section we define some properties of types, allowing us to internalize the concepts of sets, propositions and groupoids to type theory.

6.1 Contractible types

First we define the notion of contractibility. Intuitively a space is contractible if it can be deformed continuously to one of its points (e.g. a disk, $\mathbb{R}$, and so on).

Definition 11. *A proof that $X : \mathcal{U}$ is contractible consists of:*

- *A point $y : X$ called the* center of contraction.
- *For all $x : X$ a path in $x =_X y$ called the* contraction.

⚠**Warning 2.** *A naive and wrong reading of this property would be that X (as a space) is inhabited and connected (i.e. for all $x : X$ and $y : X$, we have a path from x to y). But in fact the choice of path in $x =_X y$ must depend continuously on x, making this a far stronger property. In fact this will imply that X is equivalent to $\mathbf{1}$.*

Proposition 4. *The type $\mathbf{1}$ is contractible.*

Proof. See TD1. □

In fact this example is the only one:

Proposition 5. *Let $X : \mathcal{U}$ be a contractible type, then the map $X \to \mathbf{1}$ with constant value $*$ is an equivalence.*

Proof. Assume $c : X$ is the center of contraction of X. Then we define:

$$\psi :\equiv \lambda(y : \mathbf{1}).\, c$$

$$\phi :\equiv \lambda(x : X).\, *$$

Let us prove they are inverse to each other.

- To prove that for any $x : X$ we have $\psi(\phi(x)) =_X x$, we just need to prove that:

$$(x : X) \to c =_X x$$

  But this is the definition of c being a center of contraction.

- To prove that for any $y : \mathbf{1}$ we have $\phi(\psi(y)) =_{\mathbf{1}} y$, we just need to prove that:

$$(y : \mathbf{1}) \to * =_{\mathbf{1}} y$$

  This is done by induction on y.

□

Next property is often useful.

Proposition 6. *Assume given $X : \mathcal{U}$ and $x : X$. Then the type:*

$$(y : X) \times x =_X y$$

is contractible.

Proof. We choose (x, refl_x) as a center of contraction. Then we need to show that for all $z : (y : X) \times x =_X y$ we have:

$$z = (x, \mathrm{refl}_x)$$

But by based path induction we can assume that z is (x, refl_x), in which case this is obvious. □

This result is sometimes called *contractibility of singleton*, although we will call it *local contractibility of types*. This name is reasonable if we consider intuitively $(y : X) \times x =_X y$ as the neighborhood of x in X.

6.2 Propositions

Now we want to internalize the classical idea of proposition. Recall that the propositions-as-types paradigm allows us to see any type as a *proof-relevant* proposition. This means that we need to be careful: two proofs of this proposition (i.e. two inhabitants of this type) are not necessarily equal.

What we want to internalize is the more traditional notion of *proof-irrelevant* proposition. Our approach is straightforward: we *define* a proposition as a type with at most one element.

Definition 12. *A type $X : \mathcal{U}$ is called a proposition if for all $x, y : X$ we have:*

$$x =_X y$$

Formally we define $\mathrm{isProp} : \mathcal{U} \to \mathcal{U}$ by:

$$\mathrm{isProp}(X) :\equiv (x, y : X) \to x =_X y$$

The main point is that when assuming that a proposition A holds (i.e. A is inhabited), we never need to give a name to an inhabitant in A, because all of them are equal anyway. This mimics traditional (i.e. non-proof-relevant) mathematics.

⚠**Warning 3.** *As for contractible types the naive reading "a space X is a proposition if it is path-connected" is very wrong. In fact using the excluded-middle one can show that a proposition is either empty or contractible.*

Proposition 7. *A contractible type is a proposition.*

Proof. Exercise. □

Proposition 8. *The type* $\mathbf{0}$ *is a proposition.*

Proof. We need to prove $x = y$ for all $x, y : \mathbf{0}$, but we can prove anything assuming something in $\mathbf{0}$. □

Proposition 9. *Assume given $A : \mathcal{U}$ and $P : A \to \mathcal{U}$ a family of propositions (i.e. for all $x : A$ the type $P(x)$ is a proposition). Then the type:*

$$(x : A) \to P(x)$$

is a proposition.

Proof. Assume given $f, g : (x : A) \to P(x)$, we need to prove $f = g$. By function extensionality we just need to prove that:

$$f(x) =_{P(x)} g(x)$$

for all $x : X$. But $P(x)$ is a proposition so we can conclude. □

It should be noted that A is any type, and not necessarily a proposition. If we apply this result to the case where A is a proposition, it implies that for $A, B : \mathcal{U}$ two propositions, the type $A \to B$ is a proposition. This is the traditional implication of propositions.

Proposition 10. *Assume given $A, B : \mathcal{U}$ two propositions. Then $A \times B$ is a proposition.*

Proof. We need to prove that for all $z, z' : A \times B$ we have $z =_{A \times B} z'$. By induction on z and z' it is enough to prove:

$$(x, y) =_{A \times B} (x', y')$$

for all $x, x' : A$ and $y, y' : B$. But by the computation of identity types of products we know this is equivalent to:

$$(x =_A x') \times (y =_B y')$$

and this type is inhabited because A and B are propositions. □

In the usual language $A \times B$ would be called the conjunction of A and B (meaning the property "A and B"). Note that we have a dependent version.

Proposition 11. *Assume given $A : \mathcal{U}$ a proposition and $B : A \to \mathcal{U}$ a family of propositions. Then $(x : A) \times B(x)$ is a proposition.*

Proof. We proceed as in the previous proposition, for $x, x' : A$ with $y : B(a)$ and $y' : B(a')$ we need to prove that:

$$(x, y) =_{(z:A)\times B(z)} (x', y')$$

But by the computation of identity types of dependent products we know this is equivalent to:

$$(p : x =_A x') \times (\mathbf{tr}^B_p(y) =_{B(x')} y')$$

and this type is inhabited because A and $B(x')$ are propositions. □

⚠**Warning 4.** *One should be careful that our notion of propositions is less stable than the one from usual mathematics:*

- *If $A : \mathcal{U}$ is any type, and $P : A \to \mathcal{U}$ is a family of propositions, the type*

  $$(x : A) \times P(x)$$

  is not a proposition. This is because in constructive mathematics, the existential statements are proof relevant: to prove them we need to construct an element.

- *Similarly for $A, B : \mathcal{U}$ two propositions, the type $A+B$ is not a proposition (recall that in TD1 we see that $\mathbf{1} + \mathbf{1} \simeq \mathbf{2}$ but $\mathbf{2}$ is not a proposition because we have $0 \neq_{\mathbf{2}} 1$), so it does not correspond to the traditional disjunction. Indeed to prove it constructively we need to indicate which side is true.*

We will see later how to define proof irrelevant existential and disjunctive statements.

Now we prove a first result about identity types in propositions.

Proposition 12. *Let $A : \mathcal{U}$ be a proposition. Then for all $x, y : A$, the type $x =_A y$ is contractible.*

Proof. Assume given $x, y : A$, then we can prove that A is contractible (say with center of contraction x). Therefore $A =_{\mathcal{U}} \mathbf{1}$, but we know from TD1 that identity types in $\mathbf{1}$ are equivalent to $\mathbf{1}$, so that $x =_A y$ is equivalent to $\mathbf{1}$. So $x =_A y$ is contractible. □

Recall that for $A : \mathcal{U}$ and $P : A \to \mathcal{U}$ a family of propositions, the type $(x : A) \times P(x)$ is far from a proposition. In fact we can characterize its identity types.

Proposition 13. *Assume given $A : \mathcal{U}$ and $P : A \to \mathcal{U}$ a family of propositions. For $a, a' : A$ with $p : P(a)$ and $p' : P(a')$, we have:*

$$(a, p) =_{(x:A)\times P(x)} (a', p') \simeq (a =_A a')$$

Proof. By the characterization of path types in products we have:

$$(a, p) =_{(x:A)\times P(x)} (a', p') \simeq (e : a =_A a') \times (\mathbf{tr}^P_e(p) =_{P(a')} p')$$

But since $P(a')$ is a proposition the type $\mathbf{tr}^P_e(p) =_{P(a')} p'$ is contractible hence equivalent to $\mathbf{1}$. So we have:

$$(a, p) =_{(x:A)\times P(x)} (a', p') \simeq (a =_A a') \times \mathbf{1}$$

and this is equivalent to $a =_A a'$. □

6.3 Sets

Now we want to internalize the traditional notion of sets. The idea is to represent sets as discrete spaces: we define them as types where any two paths are equal.

Definition 13. *A proof that $A : \mathcal{U}$ is a set consists of a proof that for all $x, y : A$, the type $x =_A y$ is a proposition.*

Explicitely this means that A is a set of if for all $x, y : A$ and $p, q : x =_A y$, we have $p =_{x=_A y} q$. So sets are types with proof-irrelevant identity types. We give examples of sets.

Proposition 14. *Propositions are sets.*

Proof. We know that the identity types in a propositions are contractible, so they are propositions. □

Proposition 15. *The types $\mathbf{2}$, $\mathbb{N}$ and $\mathbb{Z}$ are sets.*

Proof. See TD1. □

Now we establish some closure property of sets.

Proposition 16. *Assume $A : \mathcal{U}$ is any type and $P : A \to \mathcal{U}$ is a family of sets. Then the type:*

$$(x : A) \to P(x)$$

is a set.

Proof. Assume given $f, g : (x : A) \to P(x)$, we need to prove that $f = g$ is a proposition. But by function extensionnality:

$$(f = g) \simeq (x : A) \to f(x) =_{P(x)} g(x)$$

Since $f(x) =_{P(x)} g(x)$ is a family of proposition indexed by x, we know that:

$$(x : A) \to f(x) =_{P(x)} g(x)$$

is a proposition, so we can conclude □

It is again remarkable that A is any type, and not a set. Intuitively next proposition says that a union of sets indexed by a set is again a set.

Proposition 17. *Assume given $A : \mathcal{U}$ a set and $P : A \to \mathcal{U}$ a family of set. Then:*

$$(x : A) \times P(x)$$

is a set.

Proof. We need to show that for all $x, x' : A$ with $p : P(a)$ and $p' : P(a')$, the type:

$$(x, p) =_{(x:A)\times P(x)} (x', p')$$

is a proposition, and we know it is equivalent to:

$$(e : x =_A x') \times \mathbf{tr}_e^P(p) =_{P(x)} p'$$

But this is the product of a proposition by a family of propositions, so we can conclude. □

6.4 Groupoids

In traditional mathematics a groupoid is defined as a category where all morphisms are invertible. Now we want to internalize them to HoTT. We see them as spaces with sets of paths between points.

Definition 14. *A proof that a type $A : \mathcal{U}$ is groupoid is a proof that for all $x, y : A$ the type:*

$$x =_A y$$

is a set.

The intuition is that if $A : \mathcal{U}$ is a groupoid, then $x, y : A$ are objects of the groupoid, and the type $x =_A y$ is the set of morphisms from x to y. This definition is remarkable: all the groupoid structure (composition, identities, inverse, ...) comes from the path elimination principle.

The definition is formally similar to the definition of sets, so the following propositions are proved as for sets:

Proposition 18. *Assume given $A : \mathcal{U}$ any type, and $P : A \to \mathcal{U}$ a family of groupoids. Then:*

$$(x : A) \to P(x)$$

is a groupoid.

Proposition 19. *Assume given $A : \mathcal{U}$ a groupoid and $P : A \to \mathcal{U}$ a family of groupoids. Then:*

$$(x : A) \times P(x)$$

is a groupoid.

Note that this idea can be iterated, for example we can define a 2-groupoid as a type which identity types are groupoids, and so on.

6.5 The types Prop, Set and Gpd

Definition 15. *We denote by* Prop *(resp.* Set, Gpd*) the type of propositions (resp. sets, groupoids).*

Formally we define:

$$\mathrm{Prop} :\equiv (X : \mathcal{U}) \times \mathrm{isProp}(X)$$

And similarly for Set and Gpd. We will often say e.g. "Assume $X : \mathrm{Set}$ and $x : X$", where it is implicit that $x : X$ is in fact an inhabitant of the underlying type of X. I omit the proofs of the next lemma, which is a bit technical:

Proposition 20. *Being contractible (resp. a propositon, a set, a groupoid) is a proposition.*

Formally this means that we have an inhabitant of:

$$(X : \mathcal{U}) \to (p, q : \mathrm{isProp}(X)) \to p =_{\mathrm{isProp}(X)} q$$

and similarly for sets and groupoids. In practice it means that we never need to give a name to a proof that $X : \mathcal{U}$ is e.g. a proposition.

In the next proposition one should be careful that for $X, Y : \mathrm{Prop}$, the type

$$X =_{\mathcal{U}} Y$$

is actually a notation for the equality between the underlying types of X and Y.

Proposition 21. *Assume given* $X, Y : \mathrm{Prop}$ *(resp.* Set *or* Gpd*). Then:*

$$(X =_{\mathrm{Prop}} Y) \simeq (X =_{\mathcal{U}} Y)$$

(resp.

$$(X =_{\mathrm{Set}} Y) \simeq (X =_{\mathcal{U}} Y)$$

or the same for groupoids).

Proof. This is the characterization of identity types in $(x : A) \times P(x)$ with A any type ($\mathcal{U}$ in this case) and $P : A \to \mathrm{Prop}$ (in our case $P \equiv \mathrm{isProp}$, or the same for sets and groupoids). □

We can prove some conceptually important properties.

Proposition 22. *The type* Prop *is a set.*

Proof. By the previous proposition we know that for $A, B : \mathrm{Prop}$, $A =_{\mathrm{Prop}} B$ is equivalent to $A =_{\mathcal{U}} B$, which is equivalent to $A \simeq B$ by univalence.

So we just need to prove that $A \simeq B$ is a proposition. But this is straightforward using our previous results on propositions. □

Proposition 23. *The type* Set *is a groupoid.*

Proof. As the previous proposition. □

We could define similarly prove that Gpd is a 2-groupoids. In fact this can be iterated to define n-groupoids, and the type of n-groupoids is an $n+1$-groupoid. These n-groupoids are called n-truncated types in the HoTT book [19].

Now we study the equality types in Prop.

Proposition 24. *Assume given* $X, Y : \mathrm{Prop}$*. Then:*

$$(X =_{\mathrm{Prop}} Y) \simeq (X \leftrightarrow Y)$$

Proof. We know that $X =_{\mathrm{Prop}} Y$ is equivalent to $X =_{\mathcal{U}} Y$.

First we prove that for $X, Y : \mathrm{Prop}$, we have that $X \leftrightarrow Y$ implies $X =_{\mathcal{U}} Y$. By induction assume given $f : X \to Y$ and $g : Y \to X$. But then $f \circ g \sim \mathbf{id}_Y$ and $g \circ f \sim \mathbf{id}_X$ because X and Y are propositions, so we have $X \simeq Y$, so by univalence $X =_{\mathcal{U}} Y$.

Now we know that that $X =_{\mathrm{Prop}} Y$ and $X \leftrightarrow Y$ are propositions, so by the previous paragraph we just need to prove:

$$(X =_{\mathscr{U}} Y) \leftrightarrow (X \leftrightarrow Y)$$

The left to right direction is by path induction. The converse is (also) the previous remark. □

This property is often called *proposition extensionnality*.

6.6 Propositional truncations

Sometimes we are given a type $A : \mathscr{U}$, and we want to replace it by a proposition, a set or a groupoid. There is a universal way to do this. For propositions we define a type $|A|$ called the propositional truncation of A.

Inductive Definition 15. *Assume given a type $A : \mathscr{U}$.*

- *(Formation) We have a type $|A|$.*
- *(Introduction) For any $x : A$, we have $[x] : |A|$. Moreover $|A|$ is a proposition.*
- *(Elimination) In order to define a dependent function:*

  $$f : (x : |A|) \to P(x)$$

  with P a family of propositions, it is enough to define for all $x : A$:

  $$f([x]) :\equiv t$$

  for $t : P([x])$.

In practice the induction principle says that if we want to prove a proposition and we have assumed $|A|$, we can in fact assume A.

Remark 19. *It should be noted that the truncation $|X|$ is an example of higher inductive type with two constructors:*

$$[\_] : X \to |X|$$

and:

$$\mathrm{trunc} : (x, y : |X|) \to x =_{|X|} y$$

This differs from the HITs we have seen previously because $|X|$ appears in the type of the constructors for $|X|$. We say that $|X|$ is a recursive HIT. Recursive HITs can encode very complicated geometric constructions. In algebraic topology the (higher) truncations are defined by inductively adding a lot of cells of higher dimensions. HITs encode this process in a very compact way.

This construction allows us to remove proof relevance from any situation. As a first example, recall that being a proposition is a lot stronger than being path-connected. Indeed for $A : \mathcal{U}$ when assuming:

$$(x, y : A) \to x =_A y$$

we assume a *continuous* choice of paths, implying that if A is inhabited it is contractible. Nevertheless we can define connectedness using propositional truncation.

Definition 16. *Assume given $A : \mathcal{U}$. Then A is said connected if we have:*

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

The use of propositional truncation allows us to remove continuity. This example can be generalized quite a lot. We give some definitions:

Definition 17. *Assume given $A : \mathcal{U}$ and $P : A \to \mathrm{Prop}$. Then we can define a proposition $\exists(x : A).P(x)$ by:*

$$\exists(x : A).P(x) :\equiv |(x : A) \times P(x)|$$

Similarly for $A, B : \mathrm{Prop}$ we can define a proposition $A \vee B$ (read "A or B") by:

$$A \vee B :\equiv |A + B|$$

This allows us to use proof irrelevant existential and disjunctive statements. Note that $A \vee B$ is still not classical, but it is proof-irrelevant.

Remark 20. *We decomposed existentials and disjunctions as two constructors. This is satisfying, as we can now think of proof-relevance and intuitionism as two different issues. The picture is:*

$$(x : A) \times P(x) \to \exists(x : A).P(x) \to \neg\neg\big((x : A) \times P(x)\big)$$

and:

$$A + B \to A \vee B \to \neg\neg(A + B)$$

with no reverse implications, adding a proof-irrelevant intuitionist *version of logical connective in the middle.*

It is remarkable that this idea came from homotopy theory, illustrating the many connections between constructivism and geometry.

In fact we will refrain from using the notation $A \vee B$ because we use the symbol "$\vee$" for the wedge of types (to be defined later).

6.7 Set truncations, and beyond

We give a similar definition for sets.

Inductive Definition 16. *Assume given a type $A : \mathcal{U}$.*

- *(Formation) We have a type* $|A|_0$.
- *(Introduction) For any* $x : A$*, we have* $[x] : |A|_0$*. Moreover* $|A|_0$ *is a set.*
- *(Elimination) In order to define a dependent function:*

$$f : (x : |A|) \to P(x)$$

with P *a family of sets, it is enough to define for all* $x : A$*:*

$$f([x]) :\equiv t$$

for $t : P([x])$.

If we see $A : \mathcal{U}$ as a space, then $|A|_0$ is its set of path-connected components. This will be extremely useful to define the usual topological invariants. We can give a similar definition for groupoids by giving $|A|_1$ for $A : \mathcal{U}$ (corresponding to the fundamental groupoid of a space), and even for n-groupoids.

7 Homotopy groups and fiber sequences

In this section we define the basic invariants of algebraic topology: the homotopy groups. Our main goal is to prove the so-called *long exact sequence of homotopy groups*, and explain the advantages of its type-theoretic treatment.

7.1 Pointed types

Most traditional invariants in algebraic topology are defined on pointed spaces. We introduce pointed types, which are their counterpart in our setting.

Definition 18. *We denote by* $\mathcal{U}_*$ *the type of pointed types, i.e. we define:*

$$\mathcal{U}_* :\equiv (X : \mathcal{U}) \times X$$

So an element in $\mathcal{U}_*$ consists of a type $X : \mathcal{U}$ with $x : X$. When we have $X : \mathcal{U}_*$, we denote by X its underlying type, and we denote by $*_X : X$ its underlying point (or sometimes just $* : X$).

Definition 19. *Assume* $X, Y : \mathcal{U}_*$ *We define the type of maps from* X *to* Y*, denoted* $X \to_* Y$ *by:*

$$X \to_* Y :\equiv (f : X \to Y) \times (f(*_X) =_Y *_Y)$$

This type is considered pointed by the constant map to $*_Y$.

For $f : X \to_* Y$, we denote by f its underlying map, and we denote by $*_f$ the path in $f(*_X) =_Y *_Y$.

Now we analyse the identity types of these new types. An equality between pointed types is just an equivalence between the underlying types preserving basepoints.

Lemma 25. *Assume given $X, Y : \mathcal{U}_*$. Then $X =_{\mathcal{U}_*} Y$ is equivalent to the type of equivalence:*

$$f : X \simeq Y$$

with a path $f(*_X) = *_Y$.

Proof. By induction we just need to prove the result for $(X, *_X)$ and $(Y, *_Y)$ in $\mathcal{U}_*$. Then:

$$(X, *_X) =_{\mathcal{U}_*} (Y, *_Y) \simeq (\epsilon : X =_{\mathcal{U}} Y) \times \mathbf{tr}_\epsilon^{\lambda(X:\mathcal{U}).X}(*_X) =_Y *_Y$$

by the characterization of identity types in a dependent product. But recall that:

$$\mathbf{ua} : X =_{\mathcal{U}} Y \to X \simeq Y$$

is an equivalence. We can show:

$$\mathbf{tr}_\epsilon^{\lambda(X:\mathcal{U}).X}(*_X) =_Y \mathbf{ua}(\epsilon)(*_X)$$

by path induction on ϵ. □

An equality between maps between pointed types is a basepoint-preserving homotopy.

Lemma 26. *Assume given $X, Y : \mathcal{U}_*$ and $f, g : X \to_* Y$. Then we have that*

$$(f =_{X \to_* Y} g) \simeq (h : f \sim g) \times (h(*_X) \cdot *_g = *_f)$$

Proof. Exercise. Recall that $f \sim g :\equiv (x : X) \to f(x) =_Y g(x)$. □

7.2 Groups

First we recall a presumably well-known definition.

Definition 20. *A group G consists of:*

- *A* **set** *G.*
- *A unit $e : G$.*
- *A multiplication $\_ \cdot \_ : G \times G \to G$.*
- *An inverse map $\_^{-1} : G \to G$.*

Such that:

- *For all $x, y, z : G$ we have $x \cdot (y \cdot z) =_G (x \cdot y) \cdot z$.*
- *For all $x : G$ we have $x \cdot e = x$ and $e \cdot x = x$.*
- *For all $x : G$ we have $x \cdot x^{-1} = e$ and $x^{-1} \cdot x = e$.*

We denote Grp *the type of groups.*

Note that we identify G with its underlying set, as is traditional in mathematics.

Definition 21. *For $G, G' : \mathrm{Grp}$, a morphism of groups from G to G' consists of a map from $f : G \to G'$ such that:*

- *For all $g, g' : G$, we have:*

$$f(g \cdot g') =_{G'} f(g) \cdot f(g')$$

- *We have:*

$$f(e) = e$$

We denote by $\mathrm{Hom}_{\mathrm{Grp}}(G, G')$ the type of morphisms from from G to G'.

A morphism is called an isomorphism if its underlying map is an equivalence. We denote by $\mathrm{Iso}_{\mathrm{Grp}}(G, G')$ the type of isomorphisms from G to G'.

As usual we are interested in identity types in the type of groups.

Proposition 25. *For all $G, G' : \mathrm{Grp}$, we have:*

$$(G =_{\mathrm{Grp}} G') \simeq \mathrm{Iso}_{\mathrm{Grp}}(G, G')$$

Proof. Exercise. One needs univalence. □

Proposition 26. *The type* Grp *is a groupoid.*

Proof. Assume given G and G' two groups, we need to show that $G =_{\mathrm{Grp}} G'$ is a set. By the previous proposition it is enough to prove that $\mathrm{Iso}_{\mathrm{Grp}}(G, G')$ is a set. One can show that:

$$\mathrm{Iso}_{\mathrm{Grp}}(G, G') \simeq (f : G \to G') \times P(f)$$

for some $P : (G \to G') \to \mathrm{Prop}$. But then we know that $G \to G'$ is a set, and therefore so is $(f : G \to G') \times P(f)$. □

Definition 22. *Assume given a groupoid A with $a : A$. Then we define the group $\mathrm{Aut}_A(a)$ as $a =_A a$ with multiplication, unit and inverse defined as composition, reflexivity and inverse of paths.*

Usually we use this definition when A is Set or Grp, giving groups of permutations and automorphisms.

7.3 Homotopy groups, suspension and spheres

Definition 23. *Assume given X a pointed type. Then we define two pointed types:*

$$\Omega X :\equiv (* =_X *, \mathrm{refl}_*)$$

called the loop space of X, and:

$$\Sigma X :\equiv (\Sigma X, \mathbf{N})$$

called the suspension of X.

Note that the choice of $\mathbf{N}$ as a basepoint for ΣX is arbitrary.

Proposition 27. *Assume given X, Y two pointed types. Then:*

$$(\Sigma X \to_* Y) \simeq (X \to_* \Omega Y)$$

Proof. See TD3. □

Definition 24. *Assume given X a pointed type and $n : \mathbb{N}$, we define the n-th* homotopy groups $\pi_n(X)$ *of X by:*

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

Remark 21. *The set $\pi_0(X)$ is usually seen as pointed by $[*]$, and $\pi_n(X)$ is usually seen as a group with unit induced by reflexivity and multiplication induced by composition.*

We will see in TD3 that $\pi_n(X)$ is an abelian group for $n > 1$. In traditional algebraic topology homotopy groups are defined as homotopy classes of pointed maps out of the spheres. We make this formal.

Definition 25. *We define S^n the (pointed) n-dimensional sphere inductively on $n : \mathbb{N}$.*

- *S^0 is defined as the two-points type $\mathbf{2}$, pointed by $0 : \mathbf{2}$.*
- *S^{n+1} is defined as ΣS^n.*

Note that this definition is different than our previous one for S^1, but equivalent. Then the following proposition is straightforward.

Proposition 28. *Let X be a pointed type, then:*

$$\pi_n(X) \simeq |S^n \to_* X|_0$$

Proof. By the previous proposition we know that:

$$\pi_n(X) \equiv |\Omega^n X|_0 \simeq |S^0 \to_* \Omega^n X|_0 \simeq |\Sigma^n S^0 \to_* X|_0 \equiv |S^n \to_* X|_0$$

The first equivalence comes from the fact that for X any pointed type we have:

$$(S^0 \to_* X) \simeq X$$

□

7.4 Fiber sequences

We define an important tool of homotopy theory.

Definition 26. *Assume given a sequence of maps between pointed sets:*

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

It is a fiber sequence if for all $y : Y$ we have:

$$\epsilon_y : \mathbf{fib}_f(y) \simeq (g(y) = *)$$

with $\epsilon_{*_Y}(*_X, *_f) = *_g$.

⚠**Warning 5.** *With this definition, being a fiber sequence is not a proposition. This could be corrected.*

Any map between pointed types is part of a canonical fiber sequence.

Proposition 29. *Assume given $X, Y : \mathcal{U}_*$ with a map $g : X \to_* Y$. The canonical fiber sequence associated to g is:*

$$\mathbf{fib}_g(*) \xrightarrow{p_g}_* X \xrightarrow{g}_* Y$$

where:

- *The type $\mathbf{fib}_g(*)$ is defined as:*

$$(x : X) \times (g(x) =_Y *)$$

  pointed by $(*_X, *_g)$.

- *The map p_g is the projection to X (which obviously preserves the basepoints).*

Proof. We need to give for $x : X$ a term:

$$\epsilon_x : \mathbf{fib}_{p_g}(x) \simeq (g(x) = *)$$

But this is part of the Grothendieck correspondence. Then we need:

$$\epsilon_*((*_X, *_g), \mathrm{refl}_*) = *_g$$

but the equivalence from the Grothendieck correspondence is defined by transporting the second component $*_g$ along the third component refl_*. So we can conclude. □

In some sense any fiber sequence is equal to a canonical one, as proved by the next proposition.

Proposition 30. *Assume given $X, Y, Z : \mathcal{U}_*$ in a fiber sequence:*

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

then we have that $f = p_g$ as inhabitants of:

$$(V : \mathcal{U}_*) \times (V \to_* Y)$$

Proof. We have that:

$$X \simeq (y : Y) \times \mathbf{fib}_f(y)$$

by the Grothendieck correspondence. But then we have:

$$(y : Y) \times \mathbf{fib}_f(y) \simeq (y : Y) \times (g(y) = *)$$

by hypothesis.

We do not check that this can be extended to an equality of pointed maps to Y. □

7.5 The long fiber sequence of a map

Now we give a very important theorem of homotopy theory.

Theorem 2. *Assume given $X, Y, Z : \mathcal{U}_*$ in a fiber sequence:*

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

Then we have $\delta : \Omega Z \to_* X$ and a long fiber sequence:

$$\cdots \to_* \Omega^2 X \xrightarrow{\Omega^2 f}_* \Omega^2 Y \xrightarrow{\Omega^2 g}_* \Omega^2 Z \xrightarrow{\Omega\delta}_* \Omega X \xrightarrow{\Omega f}_* \Omega Y \xrightarrow{\Omega g}_* \Omega Z \xrightarrow{\delta}_* X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

Proof. See TD3. □

We see that fibers in homotopy type theory behave quite differently than the set-theoretic fibers, in the sense that iterating them still gives interesting information.

7.6 The long homotopy exact sequence

The long fiber sequence of a map is usually not presented to beginners in algebraic topology. Instead the long exact sequence of homotopy groups is presented. We present this result. We need to define exact sequence of pointed sets.

Definition 27. *A sequence of maps between pointed sets:*

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

is called exact if for all $y : Y$ we have:

$$\left(\exists (x : X).\, f(x) = y\right) \leftrightarrow g(y) = *$$

It is called an exact sequence of groups if X, Y, Z have a group structure with $*$ the unit, and f and g are morphisms of groups.

Being exact is weaker than being a fiber sequence, since we use $\exists (x : X).\, f(x) = y$ (recall this is defined as $|(x : X) \times f(x) = y|$). Now we want to prove that the set-truncation of a fiber sequence is an exact sequence. We need an auxiliary lemma.

Lemma 27. *Assume given $X, Y : \mathcal{U}$ with $f : X \to Y$ and $y : Y$, then we have:*

$$\exists (x : |X|_0).\, (|f|_0(x) = [y]) \simeq |(x : X) \times f(x) = y|$$

where $|f|_0 : |X|_0 \to |Y|_0$ is defined by $|f|_0([x]) :\equiv [f(x)]$.

Proof. Since both sides are propositions, it is enough to build maps both way:

- Assume given an inhabitant of $\exists(x : |X|_0).(|f|_0(x) = [y])$, we want to prove:

  $$|(x : X) \times f(x) = y|$$

  Since we want to prove a proposition, we can assume given $x : |X|_0$ such that $|f|_0(x) = [y]$, but using induction on the truncation again we can assume given $x : X$ such that $[f(x)] = [y]$.

  By our analysis of identity types in set-truncation in TD2, we this is equivalent to $x : X$ such that $|f(x) = y|$. Using induction on the truncation again, we can assume given $x : X$ such that $f(x) = y$, and we can conclude.

- Assume given an inhabitant of $|(x : X) \times f(x) = y|$, we want to prove:

  $$\exists(x : |X|_0).(|f|_0(x) = [y])$$

  By induction on the truncation we can assume given $x : X$ with $q : f(x) = y$, and then we have:

  $$([x], \mathrm{ap}_{[\_]}(q)) : (x : |X|_0) \times (|f|_0(x) = [y])$$

  so we can conclude.

□

Last proof is a good example of working with truncations: we eliminate them as much as possible.

Lemma 28. *The set truncation of a fiber sequence is exact.*

Proof. More precisely assume given a fiber sequence:

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

Then we prove that the sequence of pointed sets:

$$|X|_0 \xrightarrow{|f|_0}_* |Y|_0 \xrightarrow{|g|_0}_* |Z|_0$$

is exact, where $|f|_0$ is the map induced by f, i.e. defined by:

$$|f|_0([x]) = [f(x)]$$

So we need for all $y : |Y|_0$ that:

$$\exists(x : |X|_0).(|f|_0(x) = y) \leftrightarrow |g|_0(y) = [*]$$

Since both sides are propositions, their equivalence is itself a proposition, therefore we can assume that y is of the from $[y']$ for $y' : Y$. Then we need to prove that:

$$\exists(x : |X|_0).(|f|_0(x) = [y']) \leftrightarrow [g(y')] = [*]$$

But then we have:

$$\big([g(y')] = [*]\big) \simeq |g(y') = *| \simeq |(x : X) \times f(x) = y'|$$

And we can conclude using the previous lemma. □

Then we can prove our first classical theorem from algebraic topology.

Theorem 3. *Assume given $X, Y, Z : \mathcal{U}_*$ in a fiber sequence:*

$$X \xrightarrow{f}_* Y \xrightarrow{g}_* Z$$

Then we have a long exact sequence of pointed sets:

$$\cdots \to_* \pi_1(X) \to_* \pi_1(Y) \to_* \pi_1(Z) \to_* \pi_0(X) \to_* \pi_0(Y) \to_* \pi_0(Z)$$

This is an exact sequence sequence of groups for $n > 0$.

Proof. We see the displayed sequence of pointed sets is just the set-truncation of the long fiber sequence, so we can conclude using the previous proposition. □

An exact sequence of groups allows one to extract some information about groups in the sequence from information about the others. As an example:

Lemma 29. *Assume given an exact sequence of groups:*

$$0 \to G \xrightarrow{i} G' \to 0$$

Then i is an isomorphism of groups.

Proof. Omitted as it has nothing to do with homotopy type theory. □

This can be a useful tool to compute homotopy groups. For example we can build a fiber sequence:

$$S^1 \to_* S^3 \to_* S^2$$

usually called the Hopf fibration. From this we know that we have exact sequences of groups:

$$\pi_n(S^1) \to \pi_n(S^3) \to \pi_n(S^2) \to \pi_{n-1}(S^1)$$

for $n > 0$. But since $\pi_n(S^1) = 0$ for $n > 1$, one can conclude that:

$$\pi_n(S^3) \simeq \pi_n(S^2)$$

for $n > 2$.

Remark 22. *This presentation of the long homotopy exact sequence differs from the traditional one in algebraic topology.*

- *In algebraic topology, one restricts attention to the case where $f : X \to Y$ is a fibration. Since every map is a fibration up to equivalence, we do not need this restriction.*
- *We see that the long exact sequence of homotopy groups is actually the shadow of a more powerful (and higher-dimensional) fiber sequence. This fact is usually not pointed out in algebraic topology, at least not in an introductory class.*

Both these facts come from the reluctance from some teachers in algebraic topoplogy to use homotopy fibers rather than usual fibers. These notes[1] *by Johan Leray and Bruno Valette are a notable exception.*

[1] `https://www.math.univ-paris13.fr/~vallette/download/Homotopy%20Theories.pdf`

8 Groups as pointed connected types and ∞-groups

In this section we come back to groups and give an alternative definition for them in homotopy type theory. We will generalize this definition to ∞-groups, i.e. groups whose carriers are arbitrary types rather than sets. Note that we do not give full proofs of some key results. A key reference are the participative ongoing book called the Symmetry Book[2]. Also noteworthy is [6], focused on delooping of ∞-groups.

8.1 The main equivalence

We know that given a pointed type X it is possible to define a group $\pi_1(X)$ as:

$$|* =_X *|_0$$

If the type X happens to be a groupoid (i.e. its identity types are sets), then the truncation is unecessary. So from a pointed groupoid X it is possible to define a group $* =_X *$, denoted ΩX.

If X is not connected it is obvious that this construction looses some information: we forget everything about the connected components not containing the basepoint. We will prove that this is the only lost information.

Definition 28. *A pointed connected groupoid consists of a pointed type X such that:*

- *For all $x : X$ we have $|x = *|$, i.e. X is connected.*
- *$* =_X *$ is a set, implying that X is a groupoid.*

Definition 29. *For a pointed connected groupoid X, we define a group ΩX as:*

$$* =_X *$$

We only give a sketch of proof for the next theorem

Theorem 4. *The map Ω from pointed connected groupoids to groups is an equivalence. We denote its inverse by* B.

Proof. We want to define an inverse to Ω. Assume given a group G, then we define BG as a higher inductive type with:

- A point $* : \mathrm{B}G$, which will be the basepoint of BG.
- For any $g : G$, we add a path $p_g : * =_{\mathrm{B}G} *$.
- For any pair $g, g' : G$, we add a path:

$$h_{g,g'} : p_1 \cdot p_{g'} = p_{g \cdot g'}$$

- We have $\mathrm{refl}_* = p_e$ with e the unit of G.

[2] `https://github.com/UniMath/SymmetryBook`

- We assume that BG is a groupoid.

We do not state the elimination principle for BG. Then we need to prove that:

$$(* =_{\mathrm{B}G} *) \simeq G$$

which can be done using the encode decode method, indeed we define:

$$\mathrm{Eq}_{\mathrm{B}G} : \mathrm{B}G \to \mathrm{Set}$$

by:

$$\mathrm{Eq}_{\mathrm{B}G}(*) :\equiv G$$

$$\mathrm{ap}_{\mathrm{Eq}_{\mathrm{B}G}}(p_g)(g') = g' \cdot g$$

with the action of $h_{g,g'}$ derived from associativity of multiplication in G. Then we can show for all $x : \mathrm{B}G$ that:

$$(* =_{\mathrm{B}G} x) \simeq \mathrm{Eq}_{\mathrm{B}G}(x)$$

The other way we need to prove for X a connected pointed groupoid that:

$$\mathrm{B}(* =_X *) \simeq X$$

We omit the proof of this entirely. □

This theorem is very important, it means that we could *define* groups as pointed connected groupoids. As a space BG is a pointed connected space with one loop per element in G, such that composition of paths behaves as the multiplication in G, and with a trivial higher-dimensional structure. Now we give some examples.

Proposition 31. *We describe* BG *for some already defined groups.*

- *Assume given a groupoid A (we think mainly of* Set *and* Grp*) and $a : A$. Then we have:*

  $$\mathrm{BAut}_A(a) = (x : A) \times |x = a|$$

  pointed by $(a, [\mathrm{refl}_a])$.

- *Assume given a pointed type X. Then we have:*

  $$\mathrm{B}\pi_1(X) = \mathrm{BAut}_{|X|_1}([*]) = (x : |X|_1) \times |x = [*]|$$

  pointed by $([*], [\mathrm{refl}_{[*]}])$.

Proof.

- For the first example, we need to prove that $(x : A) \times |x = a|$ is a connected groupoid and that:

  $$(a, [\mathrm{refl}_a]) =_{(x:A)\times|x=a|} (a, [\mathrm{refl}_a]) =_{\mathrm{Grp}} \mathrm{Aut}_A(a)$$

  Since we have a propositional truncation on $x = a$, we have that:

  $$(a, p) =_{(x:A)\times|x=a|} (a', p') \simeq a =_A a'$$

So we see that:

$$(a, [\mathrm{refl}_a]) =_{(x:A)\times|x=a|} (a, [\mathrm{refl}_a]) \simeq a =_A a$$

and we can conclude because $\mathrm{Aut}_A(a)$ is defined as $a =_A a$, and the group multiplication is induced by concatenation of paths in both cases, so it is preserved.

All that is left to show is that $(x : A) \times |x = a|$ is connected. Assume given an inhabitant $z : (x : A) \times |x = a|$, we want to show that:

$$|z = (a, [\mathrm{refl}_a])|$$

Since we want to prove a proposition, we can assume $x : A$ and $q : x = a$ and try to prove:

$$(x, [q]) = (a, [\mathrm{refl}_a])$$

But this is equivalent to $x = a$, and we conclude using q.

- For the second case we just need to prove that:

$$\pi_1(X) = \mathrm{Aut}_{|X|_1}([*])$$

meaning that:

$$|* =_X *|_0 \simeq ([*] =_{|X|_1} [*])$$

But this is the characterization of identity types in $|X|_1$.

□

8.2 Reformulating morphisms

Now we have an alternative definition of groups, so we want to reformulate some notions from group theory.

Proposition 32. *For $G, G' : \mathrm{Grp}$, we have:*

$$\mathrm{Hom}_{\mathrm{Grp}}(G, G') \simeq \mathrm{B}G \to_* \mathrm{B}G'$$

Proof. Omitted. □

Intuitively this is because maps preserving the basepoint need to acts on loops, and they preserve composition of loops (and the constant path) because any map does.

Proposition 33. *Giving an exact sequence of groups:*

$$0 \to K \to H \to G \to 0$$

is the same thing as giving a fiber sequence:

$$\mathrm{B}K \to_* \mathrm{B}H \to_* \mathrm{B}G$$

Proof. Assume given a fiber sequence:

$$\mathrm{B}K \to_* \mathrm{B}H \to_* \mathrm{B}G$$

Then by the long homotopy exact sequence we have an exact sequence:

$$\pi_2(\mathrm{B}G) \to \pi_1(\mathrm{B}K) \to \pi_1(\mathrm{B}H) \to \pi_1(\mathrm{B}G) \to \pi_0(\mathrm{B}H)$$

But we can conclude because $\pi_2(\mathrm{B}G) = 0$ as $\mathrm{B}G$ is a groupoid, $\pi_0(\mathrm{B}H) = 0$ as $\mathrm{B}H$ is connected, and $\pi_1(\mathrm{B}L)$ is equal to L for any group L.

We omit the converse. □

Recall that our definition of fiber sequence was not so good, so that being a fiber sequence might not be a proposition, whereas being an exact sequence is a proposition. Therefore we do not hope for an equivalence between the two, but only for functions both way. This could be corrected by using another definition of fiber sequence.

8.3 Reformulating actions

We give a very classical definition. You should think a bit about it if you have not seen it already.

Definition 30. *An action of a group G is:*

- *A set X.*
- *A morphism of group from G to* $\mathrm{Aut}_{\mathrm{Set}}(X)$.

In this case we say that G acts on X. We denote the type of actions of G by Set^G.

We imagine that elements of G act on elements of X. This is made explicit in the following definition:

Lemma 30. *An action of G on X is equivalent to a map:*

$$\_\cdot\_ : G \times X \to X$$

such that:

- *For all $g, g' : G$ and $x : X$ we have:*

$$(g \cdot g') \cdot x = g \cdot (g' \cdot x)$$

- *For all $x : X$ we have:*

$$e \cdot x = x$$

Proof. Omitted as it has nothing to do with homotopy type theory. □

Now we characterize actions of G depending on $\mathrm{B}G$.

Proposition 34. *Let G be a group, then:*

$$\mathrm{Set}^G \simeq \mathrm{B}G \to \mathrm{Set}$$

Proof. By definition Set^G is:

$$(X : \mathrm{Set}) \times \mathrm{Hom}_{\mathrm{Grp}}(G, \mathrm{Aut}_{\mathrm{Set}}(X))$$

$$\simeq (X : \mathrm{Set}) \times \left(\mathrm{B}G \to_* \mathrm{BAut}_{\mathrm{Set}}(X)\right)$$

$$\simeq (X : \mathrm{Set}) \times \left(f : \mathrm{B}G \to (Y : \mathrm{Set}) \times |Y = X|\right) \times (f(*) = (X, [\mathrm{refl}_X]))$$

$$\simeq (X : \mathrm{Set}) \times (f : \mathrm{B}G \to \mathrm{Set}) \times \left((x : \mathrm{B}G) \to |f(x) = X|\right) \times (f(*) = X)$$

But then $f(*) = X$ imply that $|f(x) = X|$ for all $x : \mathrm{B}G$ (because $\mathrm{B}G$ is connected) and:

$$(x : \mathrm{B}G) \to |f(x) = X|$$

is a proposition therefore it is contractible. So our type is equivalent to:

$$(X : \mathrm{Set}) \times (f : \mathrm{B}G \to \mathrm{Set}) \times (f(*) = X)$$

$$\simeq \mathrm{B}G \to \mathrm{Set}$$

□

From now on we define new notions directly using $\mathrm{B}G$ rather than G.

Definition 31. *Assume given an action $X : \mathrm{B}G \to \mathrm{Set}$. Then we define the type of invariants of this action as:*

$$X^G :\equiv (x : \mathrm{B}G) \to X(x)$$

and we define the quotient of the action as:

$$X//G :\equiv (x : \mathrm{B}G) \times X(x)$$

The type X^G correspond to the set of points in $x : X(*)$ such that for all $g : G$ we have $g \cdot x = x$.

The type $X//G$ is not a set but a groupoid. In fact it is the groupoid obtained from $X(*)$ by adding a path from x to x' for every $g : G$ such that $g \cdot x = x'$. It is usually not defined in an introductory algebra class.

The usual set-quotient can be defined in HoTT as the set-truncation of $X//G$.

Remark 23. *As a follow up to our remark on quotients ill-behaved in set-theory, the set-quotient of an action of group can also be problematic.*

As an example when we have G a finite group acting on X finite there is in general no relation between the cardinals of G, of X and of the set-quotient of X by G. But there is a notion of cardinal for finite groupoids such that:

$$\mathrm{card}(X) = \mathrm{card}(G) \times \mathrm{card}(X//G)$$

where $X//G$ is the groupoid-quotient we have defined. (Puzzle unrelated to the class: Can you guess the notion of cardinal needed for finite groupoids ?)

In traditional algebra classes, you see that this relation is true when the action is free (meaning that if $g \cdot x = x$ for some x then $g = e$). In fact freeness of the action guarantees that $X//G$ is a set.

This is a somewhat general phenomenon: if you see a theorem about set-quotients of free group actions, there is probably an analogue for the groupoid-quotient of any action. We will see a much deeper example of this when we discuss principal bundles.

8.4 ∞-groups

We have given two definitions of groups. Now we want to sketch the notion of ∞-groups. Very roughly it is a group where we do not assume that the underlying type is a set. We examine the simpler example of an associative binary operation. So we ask for a type A with a binary operation $\_\cdot\_ : A \times A \to A$, such that we have a homotopy:

$$\alpha_{x,y,z} : (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

for any $x, y, z : A$. But this is not enough: indeed from α we obtain for $a, b, c, d : A$ two ways to show:

$$((a \cdot b) \cdot c) \cdot d = a \cdot (b \cdot (c \cdot d))$$

(can you find them ?). In order to obtain a well-behaved theory of associative operations, we should ask for a homotopy β between those two paths. But inductively for five elements we need to ask for a higher homotopy between two homotopies build from β, and so on. We know how to define this infinite tower in traditional homotopy theory, and the result is often called an operation *associative up to higher coherent homotopy*.

Open problem 1. *Is it possible to give a type in HoTT interpreted as operations associative up to coherent homotopy ?*

In my opinion this is the most important problem about the applicability of HoTT. It has a lot of implications concerning synthetic homotopy theory. So it seems difficult to define ∞-groups in HoTT, because the naive approach requires an operation associative up to coherent homotopy. But we have an alternative definition of groups as pointed connected groupoids, and this one is very straightforward to generalize.

Definition 32. *An ∞-group G consists of a connected pointed type* BG.

We use different notations for an ∞-group G and its underlying type BG just to mirror the situation with groups. This definition is indeed interpreted as something equivalent to the traditional definition of ∞-groups, so that if we prove a result about pointed connected types in HoTT, we can deduce a result about ∞-groups in traditional homotopy theory.

Remark 24. *From a topological group one can get an ∞-group structure on its underlying homotopy type. This group is called strict because its homotopies witnessing*

coherent associativity are all constant. There are a lot of ∞-group not equivalent to a strict one.

Now we give some definitions for ∞-groups.

Definition 33. *Assume given a type A and $a : A$, then we define the ∞-group $\mathrm{Aut}_A(a)$ by:*

$$\mathrm{BAut}_A(a) :\equiv (x : A) \times |x = a|$$

pointed by $(a, [\mathrm{refl}_a])$.

We are particularly interested in the ∞-group $\mathrm{Aut}_{\mathscr{U}}(A)$ for $A : \mathscr{U}$, which is called the ∞-group of automorphisms of A.

Definition 34. *A morphism of ∞-groups is a map of pointed sets.*

Now we define exact sequence of ∞-groups.

Definition 35. *An exact sequence of ∞-groups is a fiber sequence. More precisely an exact sequence:*

$$0 \to K \to H \to G \to 0$$

is by definition a fiber sequence:

$$\mathrm{B}K \to_* \mathrm{B}H \to_* \mathrm{B}G$$

Now we define actions.

Definition 36. *An action of an ∞-group G is an element in:*

$$X : \mathrm{B}G \to \mathscr{U}$$

We say that G acts on $X(*)$. Assume given an action $X : \mathrm{B}G \to \mathscr{U}$, then we define:

$$X^G :\equiv (x : \mathrm{B}G) \to X(x)$$

the type of invariants of X, and we define:

$$X//G :\equiv (x : \mathrm{B}G) \times X(x)$$

the quotient of the action.

Note that we just copied our definitions for groups.

9 Applications

Now that we have an alternative definition of (∞-)groups, we will exploit it to prove important theorems from algebraic topology. The main goal is to illustrate how homotopy type theory allows us to prove directly and easily these theorems, once we have these definitions of groups and ∞-groups.

9.1 Galois theorem for coverings

We define coverings.

Definition 37. *A covering of $X : \mathcal{U}$ is a map to X whose fibers are sets. For $X : \mathcal{U}$ we denote by:*

$$\mathrm{Cover}(X)$$

the type of coverings of X.

Remark 25. *In traditional algebraic topology there is a definition of coverings of a space X as well-behaved continuous maps to X with discrete fibers. Our definition is interpreted as something equivalent to the traditional one, because any map with homotopy fibers equivalent to sets is actually equivalent to a traditional covering.*

You should see covering over X as continuous family of sets indexed by X, as shown by next proposition. It is an easy variant of the Grothendieck correspondence.

Proposition 35. *Assume given a type $X : \mathcal{U}$. Then we have:*

$$\mathrm{Cover}(X) \simeq X \to \mathrm{Set}$$

It is a bit cumbersome to define and prove a counterpart to this result using topological spaces. Now we prove an important theorem of algebraic topology.

Theorem 5. *Assume given a pointed connected type X. Then we have:*

$$\mathrm{Cover}(X) \simeq \mathrm{Set}^{\pi_1(X)}$$

Recall that $\mathrm{Set}^{\pi_1(X)}$ is the type of sets with an action of $\pi_1(X)$.

Proof. We know that:

$$\mathrm{Cover}(X) \simeq X \to \mathrm{Set}$$

But then since Set is a groupoid, we have that:

$$X \to \mathrm{Set} \simeq |X|_1 \to \mathrm{Set}$$

Then it is enough to prove

$$|X|_1 \simeq \mathrm{B}\pi_1(X)$$

to conclude. But we know that:

$$\mathrm{B}\pi_1(X) \simeq (x : |X|_1) \times |x = [*]|$$

It is enough to prove $|x = [*]]$ inhabited for all $x : |X|_1$ is order to conclude, because in this case $|x = [*]|$ is contractible. We prove that $|x = [*]|$ is inhabited for any $x : |X|_1$. By induction we can assume x is the form $[x]$ but then:

$$|[x] = [*]| \simeq ||x = *|_0| \simeq |x = *|$$

by the characterisation of paths in a truncation. But $|x = *|$ is inhabited because X is connected. □

So we know a very explicit description of coverings of a type. The intuitive summary is that giving a cover of a pointed connected type is the same as giving a set (which will be the fiber of the cover over the basepoint), and an action of $\pi_1(X)$ on this set (which will be induced on the fiber by transport along paths in X). Note that in this section we were only concerned with plain groups.

Remark 26. *Usually in an introductory class about algebraic topology one often sees that subgroups of $\pi_1(X)$ corresponds to connected coverings of X. In fact connected coverings correspond to the so-called* transitive *actions of $\pi_1(X)$, which in turn correspond to subgroups of $\pi_1(X)$*

9.2 Classifying fiber bundles

We define fiber bundles.

Definition 38. *A fiber bundle with fibre $F : \mathcal{U}$ over $X : \mathcal{U}$ is a map f to X such that for all $x : X$ we have:*

$$|\mathbf{fib}_f(x) = F|$$

We denote by $\mathrm{Fib}_F(X)$ *the type of fiber bundles with fiber F over X.*

Remark 27. *In traditional topology, fiber bundles over X are defined as continuous map to X with (strict) fibers homeomorphic to F, and an additional topological condition of* local triviality. *This condition is removed entirely for us.*

Theorem 6. *For any types $X, F : \mathcal{U}$ we have:*

$$\mathrm{Fib}_F(X) \simeq X \to \mathrm{BAut}_{\mathcal{U}}(F)$$

Proof. By a variant of the Grothendieck correspondence, we have that:

$$\mathrm{Fib}_F(X) \simeq (X \to (Y : \mathcal{U}) \times |Y = F|)$$

But $\mathrm{BAut}_{\mathcal{U}}(F)$ is defined as $(Y : \mathcal{U}) \times |Y = F|$. □

We say that $\mathrm{BAut}_{\mathcal{U}}(F)$ classifies bundles with fiber F. This comes from a more general terminology: an object A is said to classify some kind of structure on types if giving such a structure on a type X is the same thing as giving a map from X to A. It is often useful to know that some kind of structure is classified by a type.

Note that the proof is extremely simple, it is essentially Grothendieck correspondence.

9.3 Classifying principal bundles

We define G-principal bundles for G an ∞-group.

Definition 39. *Assume given an ∞-group G and X a type. Then a principal G-bundle over X is a G-action P such that:*

$$P//G = X$$

We denote by $\mathrm{Bund}_G(X)$ *the type of G-principal bundles over X.*

Remark 28. *In traditional algebraic topology G-principal bundles over X for G a topological group are defined as a free and transitive continuous action with quotient X, obeying some additional topological hypothesis. In our definition, we can remove topological conditions. We can also remove the condition of freeness and transitiveness, which are here to guarantee that the strict quotient is the same as the homotopy quotient. Moreover our definition is more general as we can use non-strict ∞-groups.*

Theorem 7. *Assume given a type $X : \mathcal{U}$ and an ∞-group G. Then we have:*

$$\mathrm{Bund}_G(X) \simeq (X \to \mathrm{B}G)$$

Proof. By definition:

$$\mathrm{Bund}_G(X) :\equiv (P : \mathrm{B}G \to \mathcal{U}) \times \Big(\big((x : \mathrm{B}G) \times P(x)\big) \simeq X\Big)$$

But by the Grothendieck correspondence, we have:

$$(P : \mathrm{B}G \to \mathcal{U}) \times \Big((x : \mathrm{B}G) \times P(x) \simeq X\Big) \simeq (Y : \mathcal{U}) \times (f : Y \to \mathrm{B}G) \times \Big(\big((x : \mathrm{B}G) \times \mathbf{fib}_f(x)\big) \simeq X\Big)$$

But we know (also by the Grothendieck correspondence) that:

$$(x : \mathrm{B}G) \times \mathbf{fib}_f(x) \simeq Y$$

therefore we have:

$$(Y : \mathcal{U}) \times (f : Y \to \mathrm{B}G) \times \Big(\big((x : \mathrm{B}G) \times \mathbf{fib}_f(x)\big) \simeq X\Big) \simeq (Y : \mathrm{B}G) \times (Y \to \mathrm{B}G) \times (Y \simeq X)$$

$$\simeq (Y : \mathcal{U}) \times (Y \to \mathrm{B}G) \times (X =_{\mathcal{U}} Y) \simeq X \to \mathrm{B}G$$

so we can conclude. □

Note that the principal bundle associated to $f : X \to \mathrm{B}G$ is just $\mathbf{fib}_f : \mathrm{B}G \to \mathcal{U}$. Our last two theorems form the foundation of bundle theory. The proofs are very short, a lot shorter than using traditional methods.

9.4 Schreier theory

We give one last application of our alternative definition of groups.

Definition 40. *Assume given two ∞-groups K and G, we define an extension of G by K as:*

- *An ∞-group H.*
- *A pointed map $f : \mathrm{B}H \to_* \mathrm{B}G$ such that $\mathbf{fib}_f(*) =_{\mathcal{U}_*} \mathrm{B}K$, with $\mathbf{fib}_f(*)$ pointed by $(*, *_f)$.*

We denote by $\mathrm{Ext}(G, K)$ *the type of extensions of G by K.*

For any two ∞-groups K and G we have a trivial extension:

$$\mathrm{B}K \to_* \mathrm{B}K \times \mathrm{B}G \to_* \mathrm{B}G$$

The main intuition about group extensions is that they are "twisted" products of groups. It should be noted that if K and G happen to be groups, then $\mathrm{Ext}(K,G)$ is in fact equivalent to the type of exact sequence of groups:

$$0 \to K \to ? \to G \to 0$$

Theorem 8. *Assume given two ∞-groups K and G. Then we have:*

$$\mathrm{Ext}(G,K) \simeq \mathrm{Hom}_{\infty\mathrm{Grp}}(G, \mathrm{Aut}_{\mathscr{U}}(\mathrm{B}K))$$

Proof. We have:

$$\mathrm{Ext}(G,K) \simeq (X : \mathscr{U}_*) \times (f : X \to_* \mathrm{B}G) \times (\mathbf{fib}_f(*) =_{\mathscr{U}_*} \mathrm{B}K)$$

because we can check that this implies X connected. Now we have:

$$\mathrm{Ext}(G,K) \simeq (X : \mathscr{U}) \times (x : X) \times (f : X \to \mathrm{B}G) \times (f(x) = *) \times (\mathbf{fib}_f =_{\mathscr{U}_*} \mathrm{B}K)$$

But by the Grothendieck correspondence:

$$\mathrm{Ext}(G,K) \simeq (P : \mathrm{B}G \to \mathscr{U}) \times (y : P(*)) \times (P(*) =_{\mathscr{U}_*} \mathrm{B}K)$$

$$\simeq (P : \mathrm{B}G \to \mathscr{U}) \times (\epsilon : P(*) =_{\mathscr{U}} \mathrm{B}K) \times (y : P(*)) \times (y = \epsilon^{-1}(*))$$

$$\simeq (P : \mathrm{B}G \to \mathscr{U}) \times (P(*) =_{\mathscr{U}} \mathrm{B}K)$$

but since $\mathrm{B}G$ is connected, we know that for all $x : \mathrm{B}G$ we have $|P(x) = \mathrm{B}K|$, so:

$$\mathrm{Ext}(G,K) \simeq (P : \mathrm{B}G \to \mathscr{U}) \times ((x : \mathrm{B}G) \to |P(x) =_{\mathscr{U}} \mathrm{B}K|) \times (P(*) =_{\mathscr{U}} \mathrm{B}K)$$

$$\simeq (P : \mathrm{B}G \to (X : \mathscr{U}) \times |X = \mathrm{B}K|) \times P(*) =_{\mathscr{U}} \mathrm{B}K$$

But by definition this means:

$$\mathrm{Ext}(G,K) \simeq (P : \mathrm{B}G \to \mathrm{BAut}_{\mathscr{U}}(\mathrm{B}K)) \times (P(*) =_{\mathscr{U}} \mathrm{B}K)$$

which is what we want. □

One important point to note is that we use $\mathrm{Aut}_{\mathscr{U}}(\mathrm{B}K)$ rather than:

$$\mathrm{Aut}_{\mathscr{U}_*}(\mathrm{B}K) \ (= \mathrm{Aut}_{\infty\mathrm{Grp}}(K))$$

When K is a group, $\mathrm{BAut}_{\mathscr{U}}(\mathrm{B}K)$ is not a groupoid but rather a 2-groupoid, because $\mathrm{B}K =_{\mathscr{U}} \mathrm{B}K$ is not a set but a groupoid. This means that $\mathrm{Aut}_{\mathscr{U}}(\mathrm{B}K)$ is a 2-group and not a group. So when K and G are groups, this theorem tells us that extensions of G by K correspond to morphisms of 2-groups from G to $\mathrm{Aut}_{\mathscr{U}}(\mathrm{B}K)$.

Remark 29. *Some extensions often encountered in group theory are called semi-direct products, and they correspond precisely to morphisms of groups from G to $\mathrm{Aut}_{\mathrm{Grp}}(K)$. For us they come from composition with the morphism:*

$$\mathrm{Aut}_{\mathrm{Grp}}(K) \simeq \mathrm{Aut}_{\mathscr{U}_*}(\mathrm{B}K) \to \mathrm{Aut}_{\mathscr{U}}(\mathrm{B}K)$$

10 Spectra and cohomology

It is a well-known fact that homotopy groups are exceedingly hard to compute. In this section we describe another family of invariants called cohomology groups, which are easier to compute. Note that we will not present the methods to compute them, although they can be implemented in HoTT. A basic reference is by Cavallo [7]. The main method of computation using cellular cohomology is presented by Bucholtz and Favonia [5]. A reference for homology is given by Graham [12]. Homology is variant of cohomology significantly harder to use in our context, so we will not introduce it here.

10.1 Abelian groups and abelian ∞-groups

In this section we want to define abelian ∞-groups. We have seen that $\pi_1(\Omega^2 X)$ is abelian for any pointed type X. There is a kind of converse to this property. We omit all proofs in this section, see [6] for a thorough treatment of this (and much more).

Proposition 36. *Let A be an abelian group. There exists a pointed connected type X such that $\Omega X = \mathrm{B}A$.*

Such an X is denoted $\mathrm{B}^2 A$. The type of such X is actually contractible.

Remark 30. *This can be used to build an equivalence Ω^2 from simply connected 2-types to abelian groups. They are pointed types X such that:*

- *X is connected.*
- *$\pi_1(X)$ is the trivial group.*
- *$* =_X *$ is a groupoid, implying that X is a 2-groupoid.*

This suggests to define ∞-abelian groups as ∞-groups G that can be delooped (i.e. $\mathrm{B}G = \Omega X$ for some X). While this definition makes sense, it does not give a reasonable notion of abelian ∞-group.

Remark 31. *Deloopable ∞-groups are called* braided ∞-groups.

Next theorem is a special case of the stability theorem from [6].

Theorem 9. *Let A be an abelian group. For any natural number $n > 0$, there exists a connected type X (denoted $\mathrm{B}^n A$) such that:*

$$\Omega^n X \;=\; A$$

The goal of these result was to motivate the following definition, suggested by the fact that abelian groups can be delooped arbitrarily many times:

Definition 41. *An abelian ∞-group is a pointed connected type with infinitely many deloopings.*

More precisely it a family of pointed connected types X_n for $n : \mathbb{N}$ such that for all n we have:

$$\Omega X_{n+1} = X_n$$

This definition is satisfying, as we will illustrate in the next sections. We have mentioned that abelian groups can be seen as abelian ∞-groups.

Remark 32. *The notion of ∞-groups deloopable n times is also relevant (e.g. it gives braided ∞-groups for $n = 1$). This notion allows us to see "how much commutativity" an ∞-group has: the more it can be delooped, the more it commutes.*

10.2 Spectra and cohomology

In an introductory class to algebraic topology, cohomology theories with coefficient in an abelian group are defined. We will define them with coefficient in a spectrum, a generalisation of abelian ∞-groups.

Definition 42. *A spectrum A is a family of pointed types A_n for $n : \mathbb{Z}$ such that for all n we have:*

$$\Omega A_{n+1} = A_n$$

Note that a spectrum is not necessarily an abelian ∞-groups because we do not ask for the pointed types to be connected. The fact that we use $\mathbb{Z}$ rather than $\mathbb{N}$ is unimportant, because for $n : \mathbb{N}$ we have:

$$X_{-n} = \Omega^n X_0$$

We use it because it leads to more convenient notations. We are now ready to define the cohomology groups.

Definition 43. *Assume given a pointed type X and a spectrum A, for $n : \mathbb{Z}$ we define the abelian group $H^n(X, A)$ by:*

$$H^n(X, A) :\equiv |X \to_* A_n|_0$$

with its group structure induced by the fact that A_n is a loop space.

The abelian group $H^n(X, A)$ is called the n-th cohomology group of X with coefficient in A.

For example if A is an abelian group, then $H^n(X, A)$ is defined as

$$|X \to_* \mathrm{B}^n A|_0$$

Note that we see a duality between homotopy and cohomology groups: the former are defined as maps *from* some fixed types (more precisely from the spheres), the latter are defined as maps *to* some fixed types (more precisely to elements of a spectrum). We will see that cohomology groups have some dual properties to homotopy groups, for example homotopy interacts well with Ω, whereas cohomology interacts well with Σ.

Remark 33. *Perhaps you know about group cohomology. We define it in our context. If G is an ∞-group, then we define the cohomology of G with coefficient in a spectrum A by:*

$$H^n_{\mathrm{Grp}}(G, A) :\equiv |\mathrm{B}G \to_* A_n|_0$$

10.3 Eilenberg-Steenrod axioms for cohomology

Now we present some properties of cohomology theories, which allow us to compute them efficiently. Elilenberg and Steenrod *defined* cohomology theories using these properties.

Proposition 37. *Assume given a spectrum A, for any pointed type X and $n : \mathbb{Z}$ we have:*

$$H^n(X, A) = H^{n+1}(\Sigma X, A)$$

Proof. We have:

$$H^n(X, A) \equiv |X \to_* A_n|_0 \simeq |X \to_* \Omega A_{n+1}|_0 \simeq |\Sigma X \to_* A_{n+1}| \equiv H^{n+1}(\Sigma X, A)$$

We do not check that this respects the group structure. □

Inductive Definition 17. *Assume given a set I and a family of pointed types X_i for $i : I$. Then we define the type $\bigvee_I X_i$ by:*

- *(Formation) There is a type $\bigvee_I X_i$.*
- *(Introduction) There is a point $* : \bigvee_I X_i$. For any $i : I$ and $x : X_i$ there is a point $\mathbf{inc}_i(x) : \bigvee_I X_i$. For any $i : I$ there is a path $* = \mathbf{inc}_i(*)$.*
- *(Non-dependent Elimination) Omitted.*

Lemma 31. *Assume given a family of pointed type X_i indexed by I a set. Then:*

$$\Big(\bigvee_I X_i \to_* Y\Big) \simeq (i : I) \to (X_i \to_* Y)$$

Proposition 38. *Assume the axiom of choice for set-truncation. Then for all family of pointed types X_i indexed by a set I we have:*

$$H^n\Big(\bigvee_I X_i, A\Big) \simeq (i : I) \to H^n(X_i, A)$$

Proof. This is an easy consequence of the previous lemma. □

Cofiber sequence are dual to fiber sequence, so that they correspond to homotopy quotients. We omit the precise definition.

Proposition 39. *Assume given a spectrum A and a cofiber sequence:*

$$X \to_* Y \to_* Z$$

Then we have an exact sequence:

$$H^n(Z, A) \to H^n(Y, A) \to H^n(X, A)$$

The previous remarks combined with knowledge about cofiber sequences combine to give next proposition.

Proposition 40. *Assume given a spectrum A and a cofiber sequence:*

$$X \to_* Y \to_* Z$$

Then we have a long exact sequence:

$$\cdots \to H^{n-1}(X,A) \to H^n(Z,A) \to H^n(Y,A) \to H^n(X,A) \to H^{n+1}(Z,A) \to \cdots$$

Remark 34. *Note that in traditional algebraic topology a family of $H^n : \mathscr{U}_* \to \mathrm{Ab}$ for $n : \mathbb{Z}$ obeying the previous axioms is in fact equivalent to $H^n(\_, A)$ for some spectrum A. To my knowledge, whether this is provable in HoTT is an open problem.*

11 Conclusion

We conclude this introductory class by giving a summary and some perspectives.

11.1 Summary

In this class:

- We used homotopical ideas to revisit some logic, mainly with propositional truncation. This allowed us to decompose disjunction and existential quantifier through proof-irrelevant but intuitionist constructors.
- We introduced higher inductive types and illustrated how they allow one to model complicated objects easily. We proved that univalence determines the higher structure of the circle.
- We gave an alternative definition of groups, which sheds a new light on some basic constructions, for example quotients.
- Using Grothendieck correspondence and our alternative definition of groups, we were able to prove four important results from homotopy theory relatively easily.
- We presented cohomology and homotopy groups in a similar fashion, emphasizing the duality between the two.

The main insights we gained:

- For homotopy theorists: constructive methods and ideas can be useful for homotopy theory. This actually applies to mathematics in general, although we did not demonstrate this.
- For type theorists: Homotopical ideas are useful to understand identity types even in "normal" type theory. For example higher inductive types are useful even without univalence.

- For foundations: The idea "being isomorphic means being the same" can be taken seriously, and imply some homotopical structure on the universe !

We list some limitations to HoTT.

- HoTT is not computational. This means that when we define e.g. a natural number using univalence, it does not necessarily compute to a numeral (recall that numerals are 0, and $\mathbf{s}(n)$ for n a numeral). As an example if we show that $\pi_4(S^3) = \mathbb{Z}/n\mathbb{Z}$, there is no automated way in HoTT to compute the value of n (see [4] for more information on this). This problem can be solved using *cubical type theory* [9], which will be introduced later in this class. Some workarounds in non-cubical HoTT are being investigated under the name of *homotopy canonicity* (see [10] for homotopy canonicity in cubical type theory).

- We mentioned the idea of infinite towers of coherences when we introduced ∞-groups. It is not known whether such an infinite tower can be defined in HoTT. We were able to use a trick in order to define ∞-groups anyway, but no such trick is known for e.g. monoids. *Two-Level type theories* (e.g. [3]) are being investigated as a solution to this problem.

- HoTT is not directed, in the sense that we only say things about isomorphisms and not about morphisms. As an example we can extract the notion of isomorphisms of groups from the definition of groups in HoTT, but not the notion of morphisms of groups. Solutions to this are being investigated under the name of *directed type theories* (e.g. [16] for a two-dimensional version).

- Currently no new theorem about homotopy types was proved in HoTT. It should be noted that some theorems were extended from homotopy types to any ∞-topos using HoTT (Intuitively an ∞-topos is a mathematical universe similar to the universe of homotopy types, see below for more information).

The main thing to remember about these limitations is that HoTT is not the definitive answer to any question, but rather a beautiful first step with many ramifications.

11.2 Toward higher mathematics

I give some perspectives from the point of view of someone interested in fundamental mathematics.

Nowadays we see a tendency to replace mathematical objects by some infinite dimensional versions, for example groups become ∞-groups, abelian groups become abelian ∞-groups (or perhaps spectra). This actually can be generalized to almost anything, so that we have for example ∞-categories. Such higher dimensional objects are getting used more and more, for example they were a key ingredient in the proof of Weibel's conjecture in algebraic geometry, by Kerz, Strunk and Tamme in 2018.

Remark 35. *In fundamental physics one can study quantum fields, which describe matter and light in a space-time. Different fields on the same space-time can correspond to each other through what are called gauge equivalences, and gauge equivalent fields describe the same physical reality. But sometimes you have non-trivial gauge equivalences of a field with itself, and this fact can be observed through experiments. These are called ghost fields by physicians. Similarly you have gauge equivalences between gauge equivalences, and so on, leading to ghost-of-ghost fields, etc.*

It turns out that fields on a space-time form a genuinely ∞-dimensional object, called a smooth ∞-groupoid. Higher gauge theory *is a useful keyword if you want to learn more about this, and Urs Schreiber wrote a lot of interesting things on this idea.*

The situation is as follow:

- Objects in mathematics usually have an ∞-dimensional analogue.
- An ∞-dimensional theory should be somewhat similar to its 0-dimensional counterpart, as was worked out in details for many examples.
- But ∞-dimensional objects tend to be very difficult to work with, they can get very technical (see e.g. the famous [17]).

So we know that there is an upgraded version of basically any mathematical notion, which is quite similar to the basic version, but it requires a lot of technical work to use it. We want some techniques allowing us to manipulate the upgraded version almost as if it was the basic version, but nevertheless having these new and useful infinite dimensional features. Homotopy type theory provides that to a certain degree. A notable example in these notes is the Grothendieck correspondence: it required only a bit more work than for sets (still a bit more, but in mathematics nothing is free...), but has much more powerful consequences as illustrated by the four deep theorems from algebraic topology we deduced from it.

Homotopy type theory is proof relevant and this can be used to derive some results about homotopy types. But Homotopy type theory is also intuitionist. Is this extra generality useful? The answer is yes. Recall that families over quotients are not so well-behaved in set theory. In category theory we have a notion of *colimits* which generalize the notion of quotients. Intuitively an ∞-topos is defined as a mathematical universe where families over colimits are well-behaved (technically this is called the descent property). So sets do not form an ∞-topos, but homotopy types do. In fact any ∞-topos is quite similar to the one of homotopy types, with the notable exception that an arbitrary ∞-topos does not obey the law of the excluded middle or the axiom of choice. Mike Schulman showed that homotopy type theory can be interpreted in any ∞-topos [18]. As ∞-topoi are one of the most useful new higher dimensional objects, this allows us to amplify further the applicability of HoTT.

Remark 36. *This means that all the results we proved in this class are valid in any ∞-topos, as long as they did not use the excluded middle or choice. So for example we proved Schreier theory for ∞-groups in any ∞-topos, a result which would be so cumbersome to prove using classical methods that (to my knowledge) no such direct proof has ever been written.*

Remark 37. *Favonia, Eric Finster, Dan Licata and Peter LeFanu Lumsdaine manage to prove the Blakers-Massey theorem in HoTT [14], therefore reproving this result for homotopy types, but also generalizing it to any ∞-topos. Mathieu Anel, Georg Biedermann, Eric Finster and André Joyal derived some generalization of this result in any ∞-topos in [1], inspired by this new proof. They derive some new results about Goodwillie calculus of functors from this [2].*

The moral of this story is that homotopy type theory has proven itself a useful (albeit not definitive) tool in the development of these new higher mathematics. I hope that in the future more efficient tools using ideas from constructive mathematics will be developed, and that mathematicians will use constructivist and intuitionistic methods as a way to prove new theorems, whenever they are an appropriate tool.

References

[1] Mathieu Anel, Georg Biedermann, Eric Finster, and André Joyal. A generalized blakers-massey theorem. *arXiv preprint arXiv:1703.09050*, 2017.

[2] Mathieu Anel, Georg Biedermann, Eric Finster, and André Joyal. Goodwillie's calculus of functors and higher topos theory. *Journal of Topology*, 11(4):1100–1132, 2018.

[3] Danil Annenkov, Paolo Capriotti, and Nicolai Kraus. Two-level type theory and applications. *arXiv preprint arXiv:1705.03307*, 2017.

[4] Guillaume Brunerie. *Sur les groupes d'homotopie des sphères en théorie des types homotopiques*. PhD thesis, Nice, 2016.

[5] Ulrik Buchholtz and Kuen-Bang Hou Favonia. Cellular cohomology in homotopy type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 521–529, 2018.

[6] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. Higher groups in homotopy type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 205–214, 2018.

[7] Evan Cavallo. *Synthetic cohomology in homotopy type theory*. PhD thesis, MA thesis, 2015.

[8] Evan Cavallo and Robert Harper. Higher inductive types in cubical computational type theory. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–27, 2019.

[9] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *arXiv preprint arXiv:1611.02108*, 2016.

[10] Thierry Coquand, Simon Huber, and Christian Sattler. Homotopy canonicity for cubical type theory. *arXiv preprint arXiv:1902.06572*, 2019.

[11] Peter Dybjer. Inductive families. *Formal aspects of computing*, 6(4):440–465, 1994.

[12] Robert Graham. Synthetic homology in homotopy type theory. *arXiv preprint arXiv:1706.01540*, 2017.

[13] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. *Twenty-five years of constructive type theory (Venice, 1995)*, 36:83–111, 1998.

[14] Kuen-Bang Hou, Eric Finster, Daniel R Licata, and Peter LeFanu Lumsdaine. A mechanization of the blakers-massey connectivity theorem in homotopy type theory. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 565–574, 2016.

[15] Chris Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after voevodsky). *arXiv preprint arXiv:1211.2851*, 2012.

[16] Daniel R Licata and Robert Harper. 2-dimensional directed type theory. *Electronic Notes in Theoretical Computer Science*, 276:263–289, 2011.

[17] Jacob Lurie. *Higher topos theory*. Princeton University Press, 2009.

[18] Michael Shulman. All $(\infty, 1)$-toposes have strict univalent universes. *Preprint*, 2019.

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

[20] Benno Van Den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.