UNIVERSITÄT HEIDELBERG

Fakultät für Mathematik und Informatik

Bachelorarbeit in Mathematik

Cell Complexes in Homotopy Type Theory

Author: Chenyi Yang
Supervisor: Prof. Florent Schaffhauser
Submission Date: 30.07.2024

Disclaimer

I hereby confirm that the submitted thesis is original work and was written by me. All credit has been given when it refers to the work of others.

This thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Heidelberg, July 30th, 2024, Chenyi Yang, _____

Abstract

In this thesis we study the theory of CW-complexes in a synthetic way. We first introduced some important concepts of Homotopy Type Theory and used them to define cell complexes as a homotopical pushout. After that, we have proven some topological properties of CW-complexes and calculated the homotopy groups of some examples with methods in type theory.

Zusammenfassung

Wir untersuchen in dieser Arbeit die Theorie der CW-Komplexe synthetisch. Als Vorbereitung haben wir die Grundbegriffe in Homotopietyptheorie eingeführt. Danach werden die CW-Komplexe formal als Homotopiefaserprodukte definiert und mit Hilfe der Typtheorie haben wir einige topologischen Eigenschaften von CW-Komplexen bewiesen und die Homotopiegruppen berechnet.

Acknowledgements

Just before I entered highschool I have read some bibliographies of mathematicians, like David Hilbert. I was fascinated by the big picture of mathematics that Hilbert wanted to establish and how Gödel destroyed this dream. Though my highschool years were surrounded by competitions like olympiads for informatics, poems and anxiety from the change of my body, I had the very first interest on mathematics.

On the road of deviating to mathematics study I will mention Copper especially. He has introduced type theory to me in the first university year and encouraged me to learn more theoretic topics. I thank him for always being my best friend, supporting me all the time and being on my side.

I am grateful to all SASTer. With them I had a wonderful year and gained so many experiences on teaching and projects.

A salute to all my friends and supports from them! The energetic, rainbow guitarist LeukPoint, physicist 黑羽○ in future, always humorous Naive_Scrooge, 玄濟, a hard working idealist, Ming and so many! Only together with them I found the fantastic part of my own life. Of course, there is a place for all people who helped me in Heidelberg university and Germany. A great thank to the beautiful Heidelberg city!

I leave this paragraph to my girlfriend. In such a long time of my life she used her gentle and shining silhouette to ignite a light for me, supported my transition and gave me hopes.

Thank Florent, a great mentor! He had given many useful advices to me. Without him I could not finish this thesis.

I took a visit to Göttingen this spring. Across the foggy, quiet lake of the cemetery I saw the grave of Hilbert. Tall and dense cypress surround it. Though already old and lack of caring, people can still tell the enlightening epitaph:

Wir müssen wissen, wir werden wissen.

Contents

1 Introduction

Homotopy type theory(HoTT) is a relatively new branch of mathematics that aims to build an alternative theory for constructing and reasoning. It has a deep connection with the long existing intuitionistic logic, among them the important one is Martin-Löf type theory, which uses types instead of sets as a basis for mathematical objects. The type theory has a very nice property: Almost everything stays computable and thus can be verified by a computer. For a detailed introduction of Martin-Löf type theory, see [Mar84].

One can date the birth of Homotopy type theory back to the discovery of homotopical interpretation of the dependent types [AW09] and the univalence axiom [Voe10] around 10 years ago. HoTT provides a new geometric interpretation of equalities in classical type theory, which is generated by a single constructor $\mathrm{refl} : a = a$ but contains rich structures. By understanding all higher identities as a groupoid, this phenomenon was justified. We will cover this interpretation and the equivalence of types in Chapter 2.

Another highlight of HoTT is the higher inductive types. They allow us to define relations in inductive types. This widens the expressing ability dramatically. For example, we can now easily define the space $\mathbb{S}^n$ or even suspensions of a space. However, we shall note that finding a general syntax of higher inductive types is still a open problem in HoTT [LS20]. We will construct some useful higher inductive types in Chapter 3 as examples.

Soon after the univalence axiom was proposed, Voevodsky constructed a model in which types are interpreted as Kan simplical sets and the consistency of the univalence axiom in this model was confirmed [KL21]. It showed the potential of HoTT being a new mathematical foundation, and this foundation is at least as strong as the set theory with two inaccessible large cardinals. After the IAS special year of univalent foundations 2012-2013, a famous textbook of HoTT [Uni13] was written by over 50 collaborators. We will use this book as the main reference of Chapters 2 and 3.

HoTT shows its advantages in many mathematical fields. One of them is naturally the homotopy theory. By homotopy hypothesis [Gro21], (∞)-groupoids are (weak) homotopy equivalent to the simplical localization of topological spaces. Together with the homotopical view of types as (∞)-groupoids, we build an *intrinsic* homotopy theory upon HoTT that is independent of any realization of topological space.

We will take advantage of this in Chapter 4 and calculate some homotopy groups of a CW-complex, which serves a good start point in algebraic topology. We hope this gives the reader some ideas of how doing mathematics in HoTT looks like and how simple it actually is.

2 Type Theory

2.1 Homotopical view of equalities

Let X be a topological space and $a, b \in X$ be two points. There are paths $\varphi, \psi : I \to X$ with start point a and end point b. We can define a homotopy $H : I \to I \to X$ between φ and ψ. This can be understood as a "path over paths", we call this a 2-path. Therefore we get an analogous definition of 3-paths, namely homotopies between homotopies and so on.

In the category of topological spaces with homotopy classes as morphisms, namely the HoTop, we can thus identify two paths via the homotopies between them. This can be generalized to any higher homotopy. In this sense, k-paths serve possibly as k-equivalences between $k-1$ -paths.

To make this precise however, we need to choose our model carefully. Now let us give up constructing certain topological spaces and turn the attention to type theory.

In type theory there is an induction principle of the identity type, which says suppose we want to define a dependent function $f : \prod_{x,y:A} \prod_{p:x=y} P(x, y, p)$ for $P : \prod_{x,y:A}(x = y) \to \mathcal{U}$ a type family, it suffices to have a function $c : \prod_{x:A} P(x, x, \mathrm{refl}_x)$ and let $f(x, x, \mathrm{refl}_x) \coloneqq c(x)$.

With this axiomatic approach we can prove the symmtery and transitivity of identity types. Moreover, we know that there is a nice model satisfying this axiom, the Martin-Löf identity type with formation rule $\mathrm{refl} : \prod_{x:A}(x = x)$.

Now let $A : \mathcal{U}$ be a type in a (large enough) universe. An identity type $a = b$ with $a, b : A$ contains witnesses that two elements are equal, or indistinguishable. We now interpret a witness as a path between a and b, thus $\mathrm{refl}_a : a = a$ is a constant path in a. Inductively the identity type $p = q$ where $p, q : a = b$, freely generated by $\mathrm{refl}_p : p = p$ collects naturally all homotopies between paths from a to b and so on.

Actually we can also define path operations(eg. concatenation and reverse) via the induction principle of identity types.

Proposition 2.1.1 For a type A and $x, y, z : A$, we have

1. a function $\mathrm{inv} : (x = y) \to (y = x)$ which sends p to p^{-1} the inverse of a path.
2. a function $(\cdot) : (x = y) \to (y = z) \to (x = z)$ called the composition or concatenation of paths.

Proof. We define a type family $D : \prod_{x,y:A}(x = y) \to \mathcal{U}$ by $D(x, y, p) \coloneqq (y = x)$ and we have a map $c : \prod_{x:A} D(x, x, \mathrm{refl}_x)$ by $c(x) = \mathrm{refl}_x$. Following the induction principle, we get a function $f : \prod_{x,y:A} \prod_{p:x=y}(y = x)$ and $f(x, x, \mathrm{refl}_x) \coloneqq \mathrm{refl}_x$. We set $\mathrm{inv}(p) \coloneqq f(x, y, p)$. In other words, $\mathrm{refl}_x^{-1} \coloneqq \mathrm{refl}_x$.

Similarly by induction it suffices to give $\mathrm{refl}_x \cdot \mathrm{refl}_x$, which is of course refl_x. ∎

We have the following properties of composition, which are also proven easily via induction:

Proposition 2.1.2 For a type A and $x, y, z, w : A$, let $p : x = y, q : y = z, r : z = w$ be paths. Then we have:

1. $\mathrm{ru}_p : p = p \cdot \mathrm{refl}_y$ and $\mathrm{lu}_p : p = \mathrm{refl}_x \cdot p$.
2. $p^{-1} \cdot p = \mathrm{refl}_y$ and $p \cdot p^{-1} = \mathrm{refl}_x$.
3. $\left(p^{-1}\right)^{-1} = p$.
4. $p \cdot (q \cdot r) = (p \cdot q) \cdot r$.

Proof. By path induction we assume $y, z := x$ and $p, q, r := \mathrm{refl}_x$. Then $\mathrm{refl}_x = \mathrm{refl}_x \cdot \mathrm{refl}_x$ is inhabited with $\mathrm{refl}_{\mathrm{refl}_x}$ by Proposition 2.1.1. And $\mathrm{refl}_x^{-1} \cdot \mathrm{refl}_x = \mathrm{refl}_x$ since $\mathrm{refl}_x^{-1} = \mathrm{refl}_x$. Of course $\left(\mathrm{refl}_x^{-1}\right)^{-1} = \mathrm{refl}_x$. And finally, $\mathrm{refl}_x \cdot (\mathrm{refl}_x \cdot \mathrm{refl}_x) = (\mathrm{refl}_x \cdot \mathrm{refl}_x) \cdot \mathrm{refl}_x$ is inhabited by $\mathrm{refl}_{\mathrm{refl}_x}$. ■

This justifies our intuition of equalities as paths. And the Martin-Löf identity type will now be enriched by the path operations defined above.

Moreover, the functions between types can be regarded as functors acting on paths as morphisms. And we have following properties:

Lemma 2.1.3 (Application) Suppose $f : A \to B$ a function, then for $x, y : A$ we have a function on paths:

$$\mathrm{ap}_f : (x = y) \to (f(x) = f(y)) \tag{2.1.1}$$

Proof. By induction on $p : x = y$, it suffices to assume $y := x$ and $p = \mathrm{refl}_x$. In this case we define $\mathrm{ap}_f(\mathrm{refl}_x) := \mathrm{refl}_{f(x)}$. ■

The notation ap_f can be considered as the application of f to a path. Ambiguously we also write $f(p)$. We note that ap_f behaves homomorphically with concatenations, this is easy to calculate.

Proposition 2.1.4 Let $f : A \to B$ and $g : B \to C$ be two maps. Let $x, y, z : A$ and $p : x = y$ and $q : y = z$ be paths. Then we have

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

Lemma 2.1.5 (Transport) Let $P : A \to \mathcal{U}$ be a type family and $p : a = b$ a path of A, then p induces a function $p_* : P(a) \to P(b)$, and this defines a function $\mathrm{transport}^P(p, -) := (p)_*$

Proof. By induction on p, it suffices to consider the case $b := a$ and $p = \mathrm{refl}_a$, then we define $(\mathrm{refl}_a)_* := \mathrm{id}_{P(a)}$. ■

Topologically, the transport functor expresses a "path lifting" property.

A useful view of $P : A \to \mathcal{U}$ is seeing it as a fibration over A with the total space $\sum_{a:A} P(a)$. A dependent function $f : \prod_{a:A} P(a)$ is thus a section of P. The dependent version of Lemma 2.1.3 expresses the idea that we can compare the elements in a section along the transport.

Lemma 2.1.6 (Dependent map) Let $f : \prod_{(x:A)} P(x)$ a dependent function, then we have a map

$$\mathrm{apd}_f : \prod_{p:x=y} p_*(f(x)) = f(y) \tag{2.1.2}$$

Proof. By induction we assume $y := x$ and $p := \mathrm{refl}_x$. In this case $(\mathrm{refl}_x)_*(f(x)) = \mathrm{id}(f(x)) = f(x)$ is always inhabited by $\mathrm{refl}_{f(x)}$. ∎

We now give an important definition, which we will encounter repeatedly later.

Definition 2.1.7 A **pointed type** (A, a) is a type $A : \mathcal{U}$ and a point $a : A$, the **base point**. Denote by $\mathcal{U}_\bullet := \sum_{(A:\mathcal{U})} A$ the type of all pointed types in a universe $\mathcal{U}$.

Definition 2.1.8 For a pointed type (A, a), the **loop space** of (A, a) is the pointed type $\Omega(A, a) = ((a = a), \mathrm{refl}_a)$, also means Ω is a pointed function. we define the n**-th loop space** $\Omega^n(A, a)$ of (A, a) recursively:

$$\begin{aligned} \Omega^0(A, a) &= (A, a) \\ \Omega^{n+1}(A, a) &= \Omega^n(\Omega(A, a)) \end{aligned} \tag{2.1.3}$$

2.2 Quasi-inverses and bi-invertible maps

In classical logic we have already an equivalence of propositions. This is also true in type theory.

Definition 2.2.1 Two types A and B are said to be **logically equivalent** if there exist maps $f : A \to B$ and $g : B \to A$.

There are many ways to define the equivalence between types. The first intuition is borrowed from category theory:

Definition 2.2.2 A **homotopy** between two functions $f, g : A \to B$ is a witness of the type

$$f \sim g := \prod_{x:A} (f(x) = g(x)) \tag{2.2.1}$$

Definition 2.2.3 Let $f : A \to B$ be a map, we say that f is a **quasi-inverse** if there exists $g : B \to A$ such that $f \circ g \sim \mathrm{id}_B$ and $g \circ f \sim \mathrm{id}_A$. In this sense we define a type

$$\mathrm{qinv}(f) :\equiv \sum_{g:B\to A} (g\circ f \sim \mathrm{id}_A) \times (f\circ g \sim \mathrm{id}_B) \tag{2.2.2}$$

The type $\mathrm{qinv}(f)$ actually has more structure than only "true or false" judgement, this is not what we want since now the equivalence would not be unique.

Definition 2.2.4 A type A is said to be a **mere proposition** if $\forall a, b : A, a = b$.

Theorem 2.2.5 The type $\mathrm{qinv}(f)$ is not a mere proposition.

Sketch of proof. We prove firstly an equivalence between $\mathrm{qinv}(g)$ and $\prod_{x:X}(x = x)$ where $g : X \to Y$ if $\mathrm{qinv}(g)$ is inhabited. Then it suffices to construct a dependent function $f : \prod_{x:X}(x = x)$ which is not equal to id_X that sends x to refl_x. Then it shows $\prod_{x:X}(x = x)$ is not a mere proposition.
We choose $X :\equiv \sum_{A:\mathcal{U}} |2 = A|$ and $a :\equiv (2, |\mathrm{refl}_2|) : X$, where $|\cdot|$ is the (-1)-truncation that is defined in Section 3.4. Then we will have a non-trivial path $p : a = a$ corresponding to the equivalence $2 \simeq 2$ which sends 0_2 to 1_2 and 1_2 to 0_2. By induction on the type $\prod_{x:X}(x = x)$, we can construct a function $f : \prod_{x:X}(x = x)$ with $f(a) = p$. But $p \neq \mathrm{refl}_a$, thus $f \neq \mathrm{id}_X$, which completes the proof.
See Theorem 4.1.3 of [Uni13]. for a detailed proof of the equivalence and the induction principle. ∎
We now define a better way to demonstrate the equivalence, which is a mere proposition and show it is logically equivalent to $\mathrm{qinv}(f)$. This is done by adding a new datum in the definition of quasi inverses.

Definition 2.2.6 Let $f : A \to B$ be a map. We define types

$$\begin{aligned} \mathrm{linv}(f) &:\equiv \sum_{g:B\to A} (g\circ f \sim \mathrm{id}_A) \\ \mathrm{rinv}(f) &:\equiv \sum_{g:B\to A} (f\circ g \sim \mathrm{id}_B) \\ \mathrm{biinv}(f) &:\equiv \mathrm{linv}(f) \times \mathrm{rinv}(f) \end{aligned} \tag{2.2.3}$$

We say f is **bi-invertible** if it has both left inverse and right inverse.

Theorem 2.2.7 For any $f : A \to B$ $\mathrm{qinv}(f)$ and $\mathrm{biinv}(f)$ are logically equivalent.

Proof. Suppose we have (g, α, β) a quasi-inverse for f, then we have of course a bi-invertible map (g, α, g, β). Conversely if we have (g, α, h, β), we have a composition of homotopy

$$\begin{aligned} \gamma : h &\underset{\alpha}{\sim} g \circ f \circ h \underset{\beta}{\sim} g \\ \gamma(y) &= \alpha(h(y))^{-1} \cdot g(\beta(y)) \end{aligned} \tag{2.2.4}$$

Now define $\alpha' : h \circ f \sim \mathrm{id}_A$ by $\alpha'(x) = \gamma(f(x)) \circ \alpha(x)$ and (h, α', β) is a quasi-inverse. ∎

Theorem 2.2.8 For any $f : A \to B$, $\mathrm{biinv}(f)$ is a mere proposition.

Before proving this theorem, we need some helpful definitions and lemmas.

Definition 2.2.9 A type A is said to be **contractible** if there exists $a : A$ such that $\forall b : A, a = b$. Such an $a : A$ is called the **contractible center** of A.

If a type is contractible, then it is a mere proposition. As we can always find paths between points via concatenation of their paths connecting to the contractible center. Conversely, a mere proposition is contractible if and only if it is inhabited.

Definition 2.2.10 A map $f : A \to B$ is said to be a **half adjoint equivalence** if there is $g : B \to A$ with $\eta : g \circ f \sim \mathrm{id}_A$ and $\varepsilon : f \circ g \sim \mathrm{id}_B$ such that $\tau : \prod_{x:A} f(\eta x) = \varepsilon(fx)$. We then have a type

$$\mathrm{ishae}(f) :\equiv \sum_{g:B\to A} \sum_{\eta : g\circ f \sim \mathrm{id}_A} \sum_{\varepsilon : f\circ g \sim \mathrm{id}_B} \prod_{x:A} (f(\eta x) = \varepsilon(fx)) \tag{2.2.5}$$

Theorem 2.2.11 For any $f : A \to B$ $\mathrm{ishae}(f)$ and $\mathrm{qinv}(f)$ are logically equivalent.

Proof. We have $\mathrm{ishae}(f) \to \mathrm{qinv}(f)$ simply by forget the coherence datum τ. For the other direction, we have (g, η, ε) a quasi-inverse and construct $(g, \eta, \varepsilon', \tau)$. We define $\varepsilon'(b)$ to be the composition $\varepsilon(f(g(b)))^{-1} \cdot (f(\eta(g(b)))) \cdot \varepsilon(b)$. By naturality of η, we have

$$f(\eta(g(f(a)))) \cdot \varepsilon(f(a)) = f(g(f(\eta(a)))) \cdot \varepsilon(f(a)) = \varepsilon(f(g(f(a)))) \cdot f(\eta(a)) \tag{2.2.6}$$

Concatenate $\varepsilon(f(g(f(a))))^{-1}$ from both side and we get the homotopy $\tau(a)$. ∎

Thus $\mathrm{biinv}(f)$ and $\mathrm{ishae}(f)$ are also logically equivalent.

The definition of half adjoint equivalence reminds us of the adjoint equivalence in category theory. We also have the following familiar geometric concept.

Definition 2.2.12 The **fiber** of a map $f : A \to B$ over a point $y : B$ is

$$\mathrm{fib}_f(y) :\equiv \sum_{x:A} (f(x) = y) \tag{2.2.7}$$

Lemma 2.2.13 Suppose $f : A \to B$ is a half adjoint equivalence, then for any $y : B$, the fiber $\mathrm{fib}_f(y)$ is contractible.

Proof. Let $(g, \eta, \varepsilon, \tau) : \mathrm{ishae}(f)$. We choose $(g(y), \varepsilon y)$ to be our contraction center. Let $(x, p) : \mathrm{fib}_f(y)$, a path from $(g(y), \varepsilon y)$ to (x, p) is given by a path $\gamma : g(y) = x$ and an induced $\varepsilon y = f(\gamma) \cdot p$. Take $\gamma := g(p)^{-1} \cdot \eta x$ and this fulfils the condition via transporting along τ. ∎

Finally we can prove the main theorem of this section.

Proof of Theorem 2.2.8. If $\mathrm{biinv}(f)$ is not inhabited, then it must be equal to 0, which is a mere proposition. Now suppose it is inhabited, then by Theorem 2.2.7 it must be a quasi-inverse with $h : B \to A$ such that $h \circ f \sim \mathrm{id}_A$ and $f \circ h \sim \mathrm{id}_B$. We show $\mathrm{linv}(f)$ and $\mathrm{rinv}(f)$ are contractible, hence $\mathrm{biinv}(f)$ as a product of contractible types is contractible, thus a mere proposition by definition.

Assume functional extensionality, $\mathrm{linv}(f)$ is equivalent to $\sum_{g:B\to A}(g \circ f = \mathrm{id}_A)$. We could define a function $(- \circ f) : (B \to A) \to (A \to A)$ by sending g' to $g' \circ f$, also the pullback by f. Then $\mathrm{linv}(f) = \mathrm{fib}_{(-\circ f)}(\mathrm{id}_A)$. Now since $\mathrm{qinv}(f) \to \mathrm{ishae}(f)$ by Theorem 2.2.11, it is sufficient to give a quasi-inverse of $(- \circ f)$ to show that $(- \circ f)$ is a half adjoint equivalence and we can apply Lemma 2.2.13 to it. Define $(- \circ h) : (A \to A) \to (B \to A)$ by sending f' to $f' \circ h$. We have

$$\begin{aligned} (g' \circ f) \circ h &= g' \circ (f \circ h) = g' \circ \mathrm{id}_B = g' \\ (f' \circ h) \circ f &= f' \circ (h \circ f) = f' \circ \mathrm{id}_A = f' \end{aligned} \tag{2.2.8}$$

for any $f' : A \to A$ and $g' : B \to A$, which shows that $(- \circ h)$ is a quasi-inverse of $(- \circ f)$.

Similarly, $\mathrm{rinv}(f)$ is the fiber of $(f \circ -)$ over id_B and $(h \circ -)$ is a quasi-inverse of it and the proof is analogous. ∎

2.3 Equivalences of types

We will use bi-invertible map to define our type equivalences.

Definition 2.3.1 Two types $A, B : \mathcal{U}$ are **equivalent** if there is a bi-invertible map $f : A \to B$. We then denote by

$$(A \simeq B) := \sum_{f:A\to B} \mathrm{biinv}(f) \tag{2.3.1}$$

the type of equivalences between A and B.

By Theorem 2.2.7 we are now allowed to say two types are equivalent iff there exists a quasi-inverse pair between them.

Another useful characterization of equivalences of types is the following.

Definition 2.3.2 A map $f : A \to B$ is **contractible** if all its fibers are contractible. We have the type

$$\mathrm{hascontrFib}(f) := \prod_{y:B} \sum_{x_0:\mathrm{fib}_f(y)} \prod_{x:\mathrm{fib}_f(y)} (x_0 = x) \tag{2.3.2}$$

Proposition 2.3.3 For any map $f : A \to B$, $\mathrm{hascontrFib}(f)$ is a mere proposition.

Proof. In Theorem 2.5.6 we will prove $\mathrm{iscontr}\big(\mathrm{fib}_f(y)\big) :\equiv \sum_{x_0:\mathrm{fib}_f(y)} \prod_{x:\mathrm{fib}_f(y)} (x_0 = x)$ is a mere proposition. Now given $p, q : \mathrm{hascontrFib}(f)$, we have $\prod_{y:B} p(y) = q(y)$ by definition, thus $p \sim q$ and by functional extensionality $p = q$. ∎

Theorem 2.3.4 For any $f : A \to B$ $\mathrm{ishae}(f)$ and $\mathrm{hascontrFib}(f)$ are logically equivalent.

Proof. One direction is given by Lemma 2.2.13. For the other, see 4.4.3 of [Uni13]. ∎

Finally, there is a connection between two equivalences we have defined.

Proposition 2.3.5 Let A, B be mere propositions, then they are logically equivalent iff they are equivalent.

Proof. Clearly if they are type equivalent then they are logically equivalent. Now suppose we have $f : A \to B$ and $g : B \to A$, we need to show they are quasi-inverses. But $\forall y : B, f(g(y)) = y$ because B is a mere proposition and analogously we have $\forall x : A, g(f(x)) = x$. ∎

We have thus the first useful type equivalence.

Theorem 2.3.6 For any map $f : A \to B$ we have

$$\mathrm{biinv}(f) \simeq \mathrm{hascontrFib}(f) \tag{2.3.3}$$

2.4 The univalence axiom

So far we have worked only in a certain collection of types called universe. We can define maps betweens types in this universe. But in order to consider the universe as a whole and define maps over this universe, we need to find a "bigger" universe such that the Russell paradox does not occur. More precisely, we postulate:

Postulate 2.4.1 For each universe $\mathcal{U}_i$, there always exists a universe $\mathcal{U}_{i+1}$ such that $\mathcal{U}_i$ is a type in $\mathcal{U}_{i+1}$.

We then notice that:

Lemma 2.4.2 For types $A, B : \mathcal{U}_i$, there is a canonical function

$$\mathrm{idtoeqv} : \left(A \underset{\mathcal{U}_i}{=} B\right) \to (A \simeq B) \tag{2.4.1}$$

Proof. Consider the identity type family $\mathrm{id}_{\mathcal{U}_i} : \mathcal{U}_i \to \mathcal{U}_i$ defined in $\mathcal{U}_{i+1}$ and a path $p : A = B$, it induces a transport $p_* : A \to B$. By induction on p we have $(\mathrm{refl}_A)_* = \mathrm{id}_A$, which is an equivalence. ∎

We are giving the following axiom, originally proposed by Voevodsky, to justify our intuition that two isomorphic objects can be identified, this is already commonly used in many informal proofs nowadays.

Axiom 2.4.3 (Univalence Axiom) For any $A, B : \mathcal{U}_i$, the function Eq. (2.4.1) is an equivalence. We conclude that

$$\left(A \underset{\mathcal{U}_i}{=} B\right) \underset{\mathcal{U}_{i+1}}{\simeq} \left(A \underset{\mathcal{U}_i}{\simeq} B\right) \tag{2.4.2}$$

Remark 2.4.4 A universe is said to be **univalent** if it satisfies the univalence axiom. We assume that *all* universes are univalent.

A drawback of introducing UA as an axiom is that we have lost our good computational properties of naive type theory, which is rule-based and axiom-free. When the HoTT book was published, it was still unknown whether there is a constructive model for the axiom. This open problem is now solved via the so called cubical type theory(see [BCH19, VMA19]), that endows a computational possibility for proofs assuming univalence. For example, the proof assistant Agda with cubical extension is a realization of this theory.
For sole purposes we will however simply assume UA as an axiom for causal proofs. We name the inverse of idtoequiv as ua : $(A \simeq B) \to (A = B)$. This is useful when we see some applications of the univalence axiom in Section 3.2.

2.5 Homotopical n-types

In set theory, the equality of two elements in a set is just a logical proposition. But we have already learned that in type theory $a = b$ is also a type. We may now wonder what if the identity types of A are all mere propositions. It turns out that this is just how we define sets in type theory.

Definition 2.5.1 A type $A : \mathcal{U}$ is a set if it fulfills the following equivalent conditions:
1. (**uniqueness of identity proofs(UIP)**) $\forall a, b : A$ and $p, q : a = b$, we have $p = q$.
2. (**axiom K**) $\forall a : A$ and $p : a = a$ we have $p = \mathrm{refl}_a$.

It would be uninteresting if we could only work with sets in type theory. Luckily not every type is a set. In fact, the higher path structure of a type that we have discussed above is a generalization of sets called groupoids. This intuition is one of the important ideas in homotopy type theory.
This leads us to give an inductively defined predicate indicating whether a type is an n-groupoid. We begin the inductive base at -2, which is interpreted as a contractible type.

Definition 2.5.2 The predicate is-n-type : $\mathcal{U} \to \mathcal{U}$ for $n \geq -2$ is defined recursively as:

$$\begin{aligned} \text{is-(-2)-type} &:= \text{isContr}(X) := \sum_{x_0:X} \prod_{x:X} (x_0 = x) \\ \text{is-}n\text{-type}(X) &:= \prod_{x,y:X} \text{is-}(n\text{-1})\text{-type}\Big(x \underset{X}{=} y\Big) \end{aligned} \tag{2.5.1}$$

Example 2.5.3

- 1 is a (-2)-type, but 0 is not a (-2)-type.
- 1 is a (-1)-type. 0 is also a (-1)-type. Moreover, every (-1)-type is equivalent to either 1 or 0. This matches the intuition that (-1)-types are regarded as mere propositions.
- 0-types are also known as sets.

Remark 2.5.4 It is not hard to deduce that if $A : \mathcal{U}$ is an n-type, then it is also an $n+1$-type, thus 1 is an n-type for any $n \geq -2$. However, there exists always types that are not n-type. An easy example is therefore $\mathbb{S}^{n+1}$, which we will define later and is not an n-type.

This predicate itself is actually also a mere proposition.

Lemma 2.5.5 Let $A : \mathcal{U}$ and $P : A \to \mathcal{U}$ a type family. Suppose $\forall a : A, P(a)$ is an n-type, then so is $\prod_{a:A} P(a)$.

Proof. We proceed with induction on n. For $n = -2$. Assuming each $P(a)$ is contractible with center $p_a : P(a)$, then we choose the center of $\prod_{a:A} P(a)$ to be $\lambda a.p_a$. It is enough to show $\forall f, g : \prod_{a:A} P(a), f = g$, But for each $a : A$ we have $f(a) = g(a)$ via the path concatenation over center p_a, by functional extensionality we get $f = g$.
Suppose the lemma holds for n, i.e. $\prod_{a:A} P(a)$ is an n-type when all $P(a)$ are. And now if each $P(a)$ is an $n+1$-type, by definition we need to show $f = g$ is an n-type for any $f, g : \prod_{a:A} P(a)$. By functional extensionality it suffices to show $f \sim g$ is an n-type since $f \sim g \simeq f = g$, but it is just inductive hypothesis. ∎

Theorem 2.5.6 For $n \geq -2$ and $X : \mathcal{U}$, is-n-type(X) is a mere proposition.

Proof. We do induction on n again. For $n = -2$. Suppose we have two proofs (a, f) and (a', f') for isContr(X), where $a, a' : X$ and $f : \prod_{x:X}(a = x)$ and $f' : \prod_{x:X}(a' = x)$ respectively. By path induction on $\sum$-type, we need to find a $p : a = a'$ such that $p_*(f) = f'$. We choose $p := f(a')$. However X is contractible by any of two proofs, hence X a mere proposition. By the definition $\prod_{x:X}(a' = x)$ is a mere proposition since $a' = x$ is contractible, thus $p_*(f) = f'$ is alway inhabited.

Now suppose for any X, is-n-type(X) is a mere proposition. We need to show $\prod_{x,x':X}$ is-n-type$(x = x')$ is a mere proposition hence is-$(n+1)$-type(X) is a mere proposition. By Lemma 2.5.5 it suffices to show is-n-type$(x = x')$ is a mere proposition, but this is just the inductive hypothesis. ■

A predicate can define a subuniverse of our type universe, in the sense that we take the total space of the predicate. Precisely we have the type family $P_n : \mathcal{U} \to \mathcal{U}$, $P_n(A) :\equiv$ is-n-type(A) and the total space $\sum_{A:\mathcal{U}}$ is-n-type(A). We call this total space n-Type, which is a subtype of $\mathcal{U}$.

Lemma 2.5.7 For $n \geq -1$ and $X : \mathcal{U}$, we have $(X \to \text{is-}n\text{-type}(X)) \to \text{is-}n\text{-type}(X)$.

Proof. Let $f : X \to$ is-n-type(X) be the map. Let $x, x' : X$, $f(x)$ is a witness that X is an n-type, thus $(x = x')$ must be an $(n-1)$-type. Since x, x' are arbitrary we are done. ■

Theorem 2.5.8 For $n \geq -1$ and $X : \mathcal{U}$, X is an $(n+1)$-type if and only if $\forall x : X, x = x$ is an n -type.

Proof. The only if part is obvious. Now we want to show X is an $(n+1)$-type given $p : \prod_{x:X}$ is-n-type$(x = x)$. Let $x' : X$, we need to show $x = x'$ is an n-type. By Lemma 2.5.7 is suffices to give a map $f : (x = x') \to$ is-n-type$(x = x')$. By induction on $x = x'$ we may define $f(\text{refl}_x) :\equiv p(x)$. ■

Thus axiom K can also be generalized to characterize n-types.

Proposition 2.5.9 For $n \geq -1$ and $X : \mathcal{U}$, X is an n-type if and only if $\forall x : X, \Omega^{n+1}(X, x)$ is a (-2)-type, i.e. contractible.

Proof. see [Uni13] Theorem 7.2.9. ■

3 Higher Inductive Types

3.1 Introduction

It is well known that every free algebra is constructible as inductive types in dependent type theory. However, not every presented algebra, or equivalently speaking, algebra quotient with relations is constructible.

The higher inductive type(HIT) is a solution for this. Instead of just including generators in the definition of inductive types, it adds also relations in the form of path constructors. By our interpretation of equalities, these relations are factorized out and this gives us a quotient structure(up to homotopy).

For example, suppose we have two pointed topological spaces $(X, x_0), (Y, y_0)$, then the wedge sum $X \vee Y$ with respect to two points is defined as a higher inductive type:

- a inclusion inl : $X \to X \vee Y$

- a inclusion $\mathrm{inr} : Y \to X \vee Y$
- a 1-path $\mathrm{glue} : \mathrm{inl}(x_0) = \mathrm{inr}(y_0)$.

This can be understood as $X \sqcup Y/(x_0 = y_0)$ informally, i.e. a quotient space. We note that $X \vee Y$ comes equipped with a base point $\mathrm{inl}(x_0)$, or equivalently $\mathrm{inr}(y_0)$.

In fact, this construction is a special case of the general type colimit called pushout type. In the category of sets the pushout is a quotient structure "over" the coproduct. This again matches our intuition. We will see more examples in next section.

Though higher inductive types are very natural idea, it turns out to be quite cumbersome to find a general syntax describing it. In [LS20] a semantic of HIT was constructed with cell monads. But it is still unknown whether every syntax of HIT can be translated to match this semantic.

Another attempt is to reduce the type of HITs we are using in daily practice. For example, [Doo15] has constructed the n-truncation with natural numbers and pushouts. The construction of n-truncations we have given in Section 3.4 is recursive, which means a constructor may refer the to be constructed type itself as an argument. The biggest drawback of recursive inductive types is that it becomes extremely hard to do induction on that. We will then often use universal properties of theses types instead of the plain generated induction principles.

The pushout as a HIT is not recursive. One may even assume that every HIT can be constructed with non-recursive ones. But [LS20] presents also a counterexample which is not representable by non-recursive HITs.

We can still say something about the elimination and induction principles of higher inductive types in a rather general schemata. We will give many examples later and try to give the reader a feel of how this works.

In general, the induction principle of higher inductive type can be generated from definition using the functorial property of a map. For example, the wedge sum about have the following induction principle:

Given a type family $P : X \vee Y \to \mathcal{U}$ together with two dependent maps $g_1 : \prod_{x:X} P(\mathrm{inl}(x))$ and $g_2 : \prod_{y:Y} P(\mathrm{inr}(y))$, then for any $p : \mathrm{glue}_*(g_1(x_0)) = g_2(y_0)$ we can define a section $f : \prod_{z:X\vee Y} P(z)$ such that $f(\mathrm{inl}(x)) = g_1(x)$, $f(\mathrm{inr}(y)) = g_2(y)$ and $f_*(\mathrm{glue}) = p$.

We use the transport of path glue to gain a dependent path in the type family, this defines a term of the function f applied on the path constructor. If the path constructor has again arguments, then we need to have again a section of dependent paths instead of just p. This is what we usually do in recursive inductive types.

For higher paths there is also transport property and we can get higher dependent paths analogously. But the notation will be very complicated thus we need a method to convert higher path constructors to 1-path constructors in the definition of higher inductive types. This technique is called "hubs and spokes", see section 6.7 of [Uni13] for some details.

Lastly we define the interval as a higher inductive type. The interval is not a complicated construction, but it plays an important role in parameterizing paths in homotopy theory. The contractibility of the interval is also essential in many proofs later.

Definition 3.1.1 The (closed) interval I is defined with following constructors:
- two points $0_I, 1_I : I$
- a path $\mathrm{conn} : 0_I = 1_I$

The elimination principle of I says if we want to define a function $f : I \to B$ for any type B. We must have two points $b_0, b_1 : B$ and a path $p : b_0 = b_1$ such that $f(0_I) = b_0, f(1_I) = b_1$ and $f_*(\mathrm{conn}) = p$. This is just the parameterizing of a path in B.
The induction principle of the interval is formulated analogously. Given a type family $P : I \to U$ together with $a : P(0_I)$ and $b : P(1_I)$. Then for any $p : \mathrm{conn}_*(a) = b$ we can define a section $f : \prod_{x:I} P(x)$ such that $f(0_I) = a, f(1_I) = b$ and $f_*(\mathrm{conn}) = p$.

Lemma 3.1.2 The interval I is contractible.

Proof. We choose the center to be 0_I. By definition we need to define a $f : \prod_{x:I}(x = 0_I)$. We set $f(0_I) = \mathrm{refl}_{0_I}$ and $f(1_I) = \mathrm{conn}^{-1}$. It remains to find a path $p : \mathrm{conn}_*\left(\mathrm{refl}_{0_I}\right) = \mathrm{conn}^{-1}$. This type is equal to $\mathrm{conn}^{-1} \cdot \mathrm{refl}_{0_I} = \mathrm{conn}^{-1}$ by definition, which is inhabited by $\mathrm{ru}_{\mathrm{conn}^{-1}}$ in Proposition 2.1.2. ∎

3.2 Homotopical pushouts

We construct now a finite colimit of types, namely the pushout, which turns out to be very useful later.

Definition 3.2.1 The (homotopical) **pushout** $A \sqcup_C B$ of a span $A \overset{f}{\leftarrow} C \overset{g}{\rightarrow} B$ is defined as:
- $\mathrm{inl} : A \to A \sqcup_C B$
- $\mathrm{inr} : B \to A \sqcup_C B$
- $\mathrm{glue} : \prod_{c:C} \mathrm{inl}(f(c)) = \mathrm{inr}(g(c))$

with respect to the following commutative diagram:

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathrm{inr}} \\ A & \xrightarrow[\mathrm{inl}]{} & A \sqcup_C B \end{array}$$

The elimination rule says if we want to define a function $s : A \sqcup_C B \to D$ for any type D, we need to have:
- for each $a : A$, the term $s(\mathrm{inl}(a)) : D$

- for each $b : B$, the term $s(\mathrm{inr}(b)) : D$
- for each $c : C$, the path $s_*(\mathrm{glue}(c)) : s(\mathrm{inl}(f(c))) = s(\mathrm{inr}(g(c)))$

Many objects can be expressed as a type pushout. We give following examples:

Example 3.2.2

1. The pushout of the span $1 \leftarrow X \rightarrow 1$ is called the **suspension** ΣX of X.
2. The pushout of the span $X \overset{x_0}{\leftarrow} 1 \overset{y_0}{\rightarrow} Y$ is called the **wedge sum** $X \vee Y$ of X and Y.
3. This definition can be generalized to the indexed wedge sum. Given a index type I and a pointed type family $P : I \to \mathcal{U}_\bullet$, the pushout of the span $1 \leftarrow I \overset{P}{\rightarrow} \sum_{i:I}(X_i, x_i)$ is the **bouquet** $\bigvee_{i \in I} X_i$.
4. The pushout of the span $1 \leftarrow X \overset{f}{\rightarrow} Y$ is the **cone** or **cofiber** of $f : X \to Y$. Homotopically we have $Y/X \simeq \mathrm{cone}(f)$ the **quotient space**.
5. The classical definition of **smash product** $X \wedge Y \coloneqq (X \times Y)/(X \vee Y)$ can be then translated to $(X \wedge Y) = \mathrm{cone}(f)$ where $f : X \vee Y \to X \times Y$ is defined by $f(\mathrm{inl}(x)) = (x, y_0), f(\mathrm{inr}(y)) = (x_0, y), f(\mathrm{glue}) = \mathrm{refl}_{(x_0,y_0)}$ the inclusion.

Unlike the pushout type however, the pullback of types is directly definable in dependent type theory:

Definition 3.2.3 A **pullback** $A \times_C B$ of a diagram $A \overset{f}{\rightarrow} C \overset{g}{\leftarrow} B$ is given as:

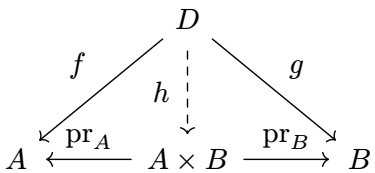
$$A \times_C B \coloneqq \sum_{a:A} \sum_{b:B} f(a) = g(b) \tag{3.2.1}$$

In general, classical type theory has already constructed the rule-based cartesian product as a type limit and coproduct as a colimit. Now with higher inductive types, finite pushout and quotient(see Section 4.1) can be included in the category of types. This makes the type category in some ways more "complete".

Now we prove some universal properties of limits and colimits in category of types.

Proposition 3.2.4 (Universal property of product type) Let $A, B : \mathcal{U}$ be any types and $A \times B : \mathcal{U}$ the product type with canonical eliminator $\mathrm{pr}_A : A \times B \to A$ and $\mathrm{pr}_B : A \times B \to B$.

1. For each $D : \mathcal{U}$ a type with $f : D \to A$, $g : D \to B$, there is a unique function $h : D \to A \times B$ making the following diagram commute (homotopically).



i.e. there are homotopies $f \sim \mathrm{pr}_A \circ h$ and $g \sim \mathrm{pr}(B) \circ g$.

2. If there is a type $C : \mathcal{U}$ together with $p : C \to A$ and $q : C \to B$ satisfying this property, then we have an equivalence $r : A \times B \simeq C$. Moreover, we get $\mathrm{ua}(r) : A \times B \underset{\mathcal{U}}{=} C$ by the univalence axiom.

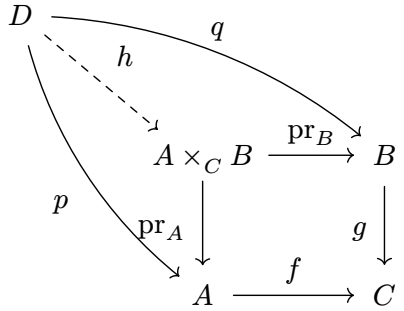
Proof. Take $d : D$, we define $h(d) \coloneqq (f(d), g(d)) : A \times B$, then we have $f(d) = \mathrm{pr}_A(h(d))$ and $g(d) = \mathrm{pr}_B(h(d))$ the expected homotopies. Further suppose we have another function h' such that $f \sim \mathrm{pr}_A \circ h'$ and $g \sim \mathrm{pr}_B \circ h'$, then $\mathrm{pr}_A \circ h' \sim \mathrm{pr}_A \circ h$ and $\mathrm{pr}_B \circ h' \sim \mathrm{pr}_B \circ h$. By the elimination principle of $A \times B$, we know for any $d : D$, $h'(d) = h(d)$ and $h \sim h'$. By functional extensionality, $h' = h$. This proves the uniqueness.
Because C and $A \times B$ all have this universal property, we get two functions $r : A \times B \to C$ and $r' : C \to A \times B$ with four homotopies:

$$\begin{aligned} p &\sim \mathrm{pr}_A \circ r' \\ q &\sim \mathrm{pr}_B \circ r' \\ \mathrm{pr}_A &\sim p \circ r \\ \mathrm{pr}_B &\sim q \circ r \end{aligned} \tag{3.2.2}$$

By $r \circ r' : C \to C$ we can get $p \sim p \circ (r' \circ r)$ and $q \sim q \circ (r' \circ r)$ via concatenation. But we know id_C has this property too so $r \circ r' = \mathrm{id}_C$ and similarly for $r' \circ r = \mathrm{id}_{A \times B}$. ∎
This is an important application of the univalence. As we have transformed the idea "isomorphic objects are the same" in the last step to a formal argumentation via ua.
The same version for pullbacks is proven analogously:

Proposition 3.2.5 (Universal property of pullback) Let $A \xrightarrow{f} C \xleftarrow{g} B$ be a diagram and $A \times_C B$ the pullback of it defined in previous definition with projections $\mathrm{pr}_A : A \times_C \to A$ and $\mathrm{pr}_B : A \times_C B \to B$.

1. For each $D : \mathcal{U}$ with $p : D \to A$, $q : D \to B$ and $s : f \circ p \sim g \circ q$, there is a unique function $h : D \to A \times_C B$ such that the following diagram commutes.



2. If there is a type $E : \mathcal{U}$ together with $i : E \to A$, $j : E \to B$ and homotopy $k : f \circ i \sim g \circ j$ satisfying this property, then we have an equivalence $r : A \times_C B \simeq E$ and $\mathrm{ua}(r) : A \times_C B = E$.

Sketch of proof. We define $h(d) \coloneqq (p(d), q(d), s(d))$ for $d : D$. The elimination principle of pullback ensures the uniqueness.

By using universal property we get $r : A \times_C B \to E$ and $r' : E \to A \times_C B$. Some computations on homotopies show they are quasi inverses. Note this can be done by any proof assistant. ■

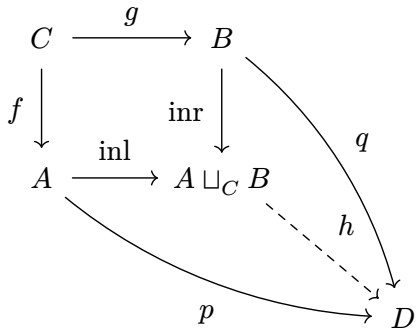
We have so far written the data of pullbacks and products in phrases. We shall remind of the definition of cone and cocone in category theory. This is not too hard to pack them together to formalize a definition:

Definition 3.2.6 Given a diagram $\mathcal{D} = A \overset{f}{\leftarrow} C \overset{g}{\to} B$ and a type D, a **cocone** of the diagram is a type $D : \mathcal{U}$ together with functions $p : A \to D$, $q : B \to D$ and a homotopy $k : p \circ f \sim q \circ g$. The type of all cocones of the diagram $\mathcal{D}$ is thus given by:

$$\operatorname{cocone}(\mathcal{D}) \coloneqq \sum_{D:\mathcal{U}} \sum_{p:A\to D} \sum_{q:B\to D} \prod_{c:C} p(f(c)) = q(g(c)) \tag{3.2.3}$$

We see that the type $\operatorname{cocone}(\mathcal{D})$ for any diagram $\mathcal{D}$ is actually inhabited by our just defined higher inductive type $A \sqcup_C B$ and its constructors. Bearing this definition in mind, we can give the universal property of pushout compactly.

Proposition 3.2.7 (Universal property of pushout) Let $\mathcal{D} = A \overset{f}{\leftarrow} C \overset{g}{\to} B$ be a diagram, then for any $D : \operatorname{cocone}(\mathcal{D})$, there is a unique function $h : A \sqcup_C B \to D$ such that the following diagram commutes.



Moreover, $A \sqcup_C B$ is a universal cocone. That is to say, for any other $E : \operatorname{cocone}(\mathcal{D})$ satisfying this property, we have $r : A \sqcup_C B \simeq E$ and $\operatorname{ua}(r) : A \sqcup_C B = E$.

Proof. With some abuse of languages let $D = (D, p, q, k)$ be a cocone. We define h with elimination principle of pushout: $h((\operatorname{inl}(a))) \coloneqq p(a), h(\operatorname{inr}(b)) \coloneqq q(b)$ and $h_*(\operatorname{glue}(c)) \coloneqq k(c)$. Now suppose we have $h' : A \sqcup_C B \to D$ such that $p = h' \circ \operatorname{inl}$ and $q = h' \circ \operatorname{inr}$ and $h' \circ \operatorname{inl} \circ f = h' \circ \operatorname{inr} \circ g$, then this formulates exactly the elimination principle hence $h' = h$.

We leave out the proof of universal since they are just tedious computations. ■

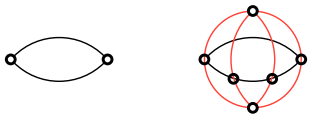
Note that the universal property of pushout is a "mapping out" property, thus it is already given by the generated elimination principle of $A \sqcup_C B$. But this is not the case for product and pullback, where they can be characterized either by a "mapping in" universal property or a "mapping out" elimination rule.

3.3 Suspensions and spheres

We have mentioned in the previous section that the suspension of a type X is a homotopical pushout $1 \leftarrow X \rightarrow 1$. We are going to prove some properties of the suspension.

Definition 3.3.1 For any type A the suspension ΣA is a higher inductive type presented by:

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



$\mathbb{S}^1$ and its suspension

We can see that because of $(1 \to A) \simeq A$, this definition coincidences with the pushout of span $1 \leftarrow A \rightarrow 1$, where we rename one 1 as N and the other as S. Again we formulate the recursive principle of ΣA since it is frequently used in the proof. In order to define a function $f : \Sigma A \to B$, it suffices to define:

- the term $f(N) : B$
- the term $f(S) : B$
- for each $a : A$, the path $f_*(\mathrm{merid}(a)) : f(N) = f(S)$

The suspension lifts a point to a path connecting N and S, and each path can be seen as an interval, which is contractible. This observation is essential in the following proposition.

Proposition 3.3.2 Let (A, a_0) and (B, b_0) be two pointed types, denote by $(A, a_0) * \to (B, b_0) :\equiv \sum_{f:A\to B}(f(a_0) = b_0)$ the type of all pointed maps from A to B, then we have

$$((\Sigma A, N) * \to (B, b_0)) \simeq ((A, a_0) * \to \Omega(B, b_0)) \tag{3.3.1}$$

Proof. Suppose we have now a map $g : A \to (b_0 = b_0)$ such that $g(a_0) = \mathrm{refl}_{b_0}$, then we can define $f : \Sigma A \to B$ by $f(N) :\equiv b_0 =: f(S)$ and $f_*(\mathrm{merid}(a)) = g(a)$. This map is indeed pointed since $f(N) = b_0$. This defines a map $\psi : ((A, a_0) * \to \Omega(B, b_0)) \to ((\Sigma A, N) * \to (B, b_0))$.
Conversely if we have a map $f : \Sigma A \to B$ such that $f(N) :\equiv b_0$. We construct $g : A \to (b_0 = b_0)$ by sending $a : A$ to $f_*(\mathrm{merid}(a))$, by Lemma 3.1.2 the type $N = S$ is contractible and we get $\mathrm{contr} : \prod_{q:N=S} q = \mathrm{refl}_N$. Calculate $f_*(\mathrm{contr}(\mathrm{merid}(a_0)))$ gives us $p : g(a_0) = \mathrm{refl}_{b_0}$ and g is pointed. This defines a map $\varphi : ((\Sigma A, N) * \to (B, b_0)) \to ((A, a_0) * \to \Omega(B, b_0))$.
We have then $\psi(\varphi(f)) = f$ and $\varphi(\psi(g)) = g$ just by the construction. ∎

Following the already known CW-structure we can define a circle $\mathbb{S}^1$ as

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

We take this approach further to define $\mathbb{S}^2$ by

- a point base : $\mathbb{S}^{\{2\}}$
- a 2-path loop : $\mathrm{refl}_{\mathrm{refl}_{\mathrm{base}}} = \mathrm{refl}_{\mathrm{refl}_{\mathrm{base}}}$

This soon encounters problems since the definitions are not inductively structured. Operating on higher paths is also an inconvenient part of this definition. In order to convert these higher paths into 1-path, we use suspensions.

Definition 3.3.3 The n-sphere $\mathbb{S}^n$ is defined recursively by:

$$\begin{aligned} \mathbb{S}^0 &:= 2 := 1 + 1 \\ \mathbb{S}^{n+1} &:= \Sigma\mathbb{S}^n \end{aligned} \tag{3.3.2}$$

This definition is much better than our naive attempt. One advantage is that we can prove the universal property of $\mathbb{S}^n$, which links our definition of loop spaces with the classical one.

Proposition 3.3.4 Let (B, b_0) be a pointed type, we have following equivalence:

$$(\mathbb{S}^n * \to B) \simeq \Omega^n(B, b_0) \tag{3.3.3}$$

Proof. By Proposition 3.3.2 and Definition 3.3.3, we can inductively transform the first term into $(1 + 1 * \to \Omega^n B)$. It remains to show $(1 + 1 * \to B) \simeq B$. First note that $(1 \to B) \simeq B$ since 1 is initial. We show $(1 + 1 * \to B) \simeq (1 \to B)$.
Without loss of generality we assume $1 + 1$ is pointed by $\mathrm{inr}(*)$. Now given a $f : 1 + 1 * \to B$ with $p : f(\mathrm{inr}(*)) = b_0$, we have $f \circ \mathrm{inl} : 1 \to B$. And given $g : 1 \to B$ we define $g' : 1 + 1 \to B$ by induction on coproduct, sending $g'(\mathrm{inl}(*)) := g(*)$ and $g'(\mathrm{inr}(*)) := b_0$, then g' is pointed.
Compute $(f \circ \mathrm{inl})'$ gives us f back and $g' \circ \mathrm{inl}$ is just g by definition. ∎
As an application we construct the real projective space $\mathbb{RP}^n$.

Lemma 3.3.5 There exists a non-trivial antipodal map $f_n : \mathbb{S}^n \to \mathbb{S}^n$.

Proof. For the case when $n = 0$, we can define $f(0) := 1$ and $f(1) := 0$ by the induction principle of 2. Now inductively, for $\mathbb{S}^n = \Sigma\mathbb{S}^{n-1}$, we do induction on the suspension. Suppose we have already a map $f_{n-1} : \mathbb{S}^{n-1} \to \mathbb{S}^{n-1}$ and we define $f_n : \mathbb{S}^n := \Sigma\mathbb{S}^{n-1} \to \Sigma\mathbb{S}^{n-1}$ by $f(N) := S$, $f(S) := N$ and $\forall x : \mathbb{S}^{n-1}, f_*(\mathrm{merid}(x)) := \mathrm{merid}(f_{n-1}(x))^{-1} : S = N$. ∎

Definition 3.3.6 The **real projective space** $\mathbb{RP}^n$ is defined by $\mathrm{cone}(f_n)$ where f_n is the one in previous lemma.

This definition is elegant but ill-behaved when calculating the property of spaces later. The reason is that the antipodal map is inductively constructed. A better solution is then to define a universal bundle of $\mathbb{RP}^n$ with the space simultaneously via inductive pushouts. [BR17] used this idea and proved an equivalence of higher homotopy groups between real projective spaces and spheres.

3.4 Type truncations

Now We want to define a general truncation of types into an n-type, or in the language of homotopy theory, its nth Postnikov section.

Definition 3.4.1 For $n \geq -1$, the truncated type $|A|_n$ of A is defined as a higher inductive type with following constructors.

- a function $|-|_n : A \to |A|_n$
- a function $h : (\mathbb{S}^{n+1} \to |A|_n) \to |A|_n$
- a dependent path $s : \prod_{r:\mathbb{S}^{n+1}\to |A|_n} \prod_{x:\mathbb{S}^{n+1}} (r(x) = h(r))$

We must convince ourself that after truncation $|A|_n$ is already an n-type.

Theorem 3.4.2 For $n \geq -1$ and $A : \mathcal{U}$, $|A|_n$ is an n-type.

Proof. By Proposition 2.5.9 we need to show $\Omega^{n+1}(|A|_n, x)$ is contractible for every $x : |A|_n$. By Proposition 3.3.4 this is equivalent to the type $(\mathbb{S}^{n+1} * \to (|A|_n, x))$.
We choose the contractible center to be c_x the constant map to x and the path $\mathrm{refl}_x : c_x(N) = x$, this is of course pointed. Now for a pair (r, p) with $r : \mathbb{S}^{n+1} \to |A|_n$ and $p : r(N) = x$, we need to show $r(y) = c_x(y) = x$. This is given by the path $s(r)(y) \cdot s(r)(N)^{-1} \cdot p$. Finally the path p after transporting along $r = c_x$ should be the path refl_x. By definition this is $\left(s(r)(N) \cdot s(r)(N)^{-1} \cdot p\right)^{-1} \cdot p = \mathrm{refl}_x$ by computation. ∎
The induction principle of n-truncation says that suppose we want to define a section $f : \prod_{x:\ |A|_n} P(x)$ for a fibration $P : |A|_n \to \mathcal{U}$, we need to have

- For each $a : A$, an element $p_a : P(|a|_n)$ such that $f(|a|_n) = p_a$
- For each $r : \mathbb{S}^{n+1} \to |A|_n$ and $r' : \prod_{x:\mathbb{S}^{n+1}} P(r(x))$, an element $h'(r, r') : P(h(r))$
- For each $r : \mathbb{S}^{n+1} \to |A|_n$, $r' : \prod_{x:\mathbb{S}^{n+1}} P(r(x))$ and $x : \mathbb{S}^{n+1}$, a dependent path $s(r)(x)_*(r'(x)) = h'(r, r')$

This principle is cumbersome to handle since we have defined here first time a recursive higher inductive type. Nevertheless, if we can ensure that the fibration lies all on n-types, then things become much simpler.

Proposition 3.4.3 Let $P : |A|_n \to \mathcal{U}$ be a fibration such that all $P(x)$ is an n-type. Then given $g : \prod_{a:A} P(|a|_n)$ there is always a section $f : \prod_{x:\ |A|_n} P(x)$ such that $f(|a|_n) = g(a)$.

Proof. The idea is to use the n-truncation to construct the second and third data listed in the induction principle above, see 7.3.2 of [Uni13] for a detailed proof. ∎
In fact, if P is a constant family to E for E an n-type, we can extend then every function $f : A \to E$ to exactly an $\mathrm{ext}(f) : |A|_n \to E$ such that $\mathrm{ext}(f)(|a|_n) = f(a)$. This is just the elimination principle of the truncation.

Proposition 3.4.4 Let $A, B : \mathcal{U}$ and B is an n-type for $n \geq -2$, then we have following equivalence

$$(A \to B) \simeq (|A|_n \to B) \tag{3.4.1}$$

Proof. For every $f : A \to B$ we have already $\mathrm{ext}(f) : |A|_n \to B$. Now suppose we have $g : |A|_n \to B$, we have a function $g \circ |-|_n : A \to B$. Moreover $\mathrm{ext}(f) \circ |-|_n = f$ by definition and $\mathrm{ext}(g \circ |-|_n)(|a|_n) = g(|a|_n)$ for every $|a|_n : |A|_n$. ∎

This is a kind of the universal property of the n-truncation. We can consider the analogy that the n-th homotopy group of a CW-complex is isomorphic to the one of the $n+1$-th section of this CW-structure. We will discuss more in next chapter.

A notable fact of n-types is that they build a reflexive subcategory of the category of types. Thus we have the following proposition immediately.

Proposition 3.4.5 A type $A : \mathcal{U}$ is an n-type if and only if $|-|_n : A \to |A|_n$ is an equivalence.

Proof. Since n-types is closed under equivalences so "if" part is clear. Conversely if A is an n -type, we have $\mathrm{ext}(\mathrm{id}_A) : |A|_n \to A$ such that $\mathrm{ext}(\mathrm{id}_A) \circ |-|_n = \mathrm{id}_A$. By Proposition 3.4.4 we only need to show $|-| \circ \mathrm{ext}(\mathrm{id}_A) \circ |-|_A = \mathrm{id}_{|A|_n} \circ |-|_n$ and use the bijectivity. This is true by computation. ∎

A dual concept of n-types in homotopy theory is the connectedness. A type A is n-connected if it has no non-trivial homotopy group below n. We define this now formally.

Definition 3.4.6 A function $f : A \to B$ is said to be n-**connected** if for any $b : B$, the n-truncated fibration $|\mathrm{fib}_f(b)|_n$ is contractible. A type $A : \mathcal{U}$ is said to be n-connected if the canonical function $A \to 1$ is n-connected, i.e. $|A|_n$ is contractible.

Proposition 3.4.7 if $A : \mathcal{U}$ is n-connected, then there is an equivalence $(A \to B) \simeq B$ for B an n -type.

Proof. By Proposition 3.4.4, we have $(A \to B) \simeq (|A|_n \to B)$, but $|A|_n \simeq 1$ because A is n-connected, thus $(A \to B) \simeq (1 \to B) \simeq B$. ∎

4 Mathematics

4.1 Algebraic structures

We construct at first another finite colimit of sets, the quotient. We need to use the truncation defined in the last section.

Definition 4.1.1 Let $A : \mathcal{U}$ be a set and $R : A \times A \to$ -1-Type a relation. The set quotient A/R is the higher inductive type generated by:

- A projection $\pi : A \to A/R$
- A function equ : $\prod_{x,y:A} R(x,y) \to (\pi(x) = \pi(y))$
- The 0-truncation trunc : $\prod_{x,y:A} \prod_{r,s:x=y} (r = s)$

The reason why we need the last truncation constructor is that in general the colimit of n-types is not an n-type anymore. In order to get a 0-type again, we need to truncate it.

After truncation we will see that the projection $\pi : A \to A/R$ is surjective, i.e. there merely exists a $x : A$ for any $q : A/R$ such that $\pi(x) = q$. But by induction on A/R, in the case where $q :\equiv \pi(x)$, we have of course $\pi(x) = \pi(x)$ and the path properties we need to verify become trivial since the goal is a mere proposition.

We have the following universal property by quotients:

Proposition 4.1.2 For any set B, we have an equivalence:

$$(A/R \to B) \simeq \sum_{f:A\to B} \prod_{x,y:A} R(x,y) \to (f(x) = f(y)) \tag{4.1.1}$$

Proof. The recursion principle of A/R says suppose we have a function $f : A \to B$ and a proof that $\prod_{x,y:A} R(x,y) \to (f(x) = f(y))$, then we have a function $f' : A/R \to B$ such that $f'(\pi(x)) = f(x)$. This defines a function $\psi : f \mapsto f'$. On the other hand, given a function $g : A/R \to B$, we have a function $g \circ \pi : A \to B$.

We have $f' \circ \pi = f$ by definition, which shows ψ is a right inverse of $(- \circ \pi)$. In order to show it is also a left inverse, i.e. $(g \circ \pi)'(q) = g(q), \forall q : A/R$, we can use the fact that π is a surjection and we may write $q = \pi(x), x : A$. In this case we have

$$(g \circ \pi)'(q) = (g \circ \pi)'(\pi(x)) = (g \circ \pi)(x) = g(q) \tag{4.1.2}$$

This completes the proof. ∎

The next important topic we will cover is the group. In general, a group is considered as a quotient of the free group with many relations. Before we see some concrete examples, lets review the basic definition of a group.

Definition 4.1.3 A **group** is a tuple $\left(G, \cdot, e, (\cdot)^{-1}\right)$ with G : 0-Type together with proofs of following conditions:

- associativity: $\prod_{x,y,z:G} x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
- inverse: $\prod_{x:G} x \cdot x^{-1} = e$ and $\prod_{x:G} x^{-1} \cdot x = e$.
- unit: $\prod_{x:G} x \cdot e = x$ and $\prod_{x:G} e \cdot x = x$.

A group is called **abelian** if it fulfills the additional condition:

- commutativity: $\prod_{x,y:G} x \cdot y = y \cdot x$.

For simplicity we will use the base set G to represent the whole group.

Definition 4.1.4 The **free group** $F(A)$ generated by a given set A is a higher inductive type generated by:

- a map $\mathrm{gen} : A \to F(A)$ which embeds all generators
- a operator $(\cdot) : F(A) \times F(A) \to F(A)$
- a unit element $e : F(A)$
- a map $(\cdot)^{-1} : F(A) \to F(A)$
- a dependent map $\mathrm{asso} : \prod_{x,y,z:F(A)} x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- two dependent maps $\mathrm{linv} : \prod_{x:F(A)} x \cdot x^{-1} = e$ and $\mathrm{rinv} : \prod_{x:F(A)} x^{-1} \cdot x = e$
- two dependent maps $\mathrm{lu} : \prod_{x:F(A)} x \cdot e = x$ and $\mathrm{ru} : \prod_{x:F(A)} e \cdot x = x$
- a truncation $\mathrm{trunc} : \prod_{x,y:F(A)} \prod_{p,q:x=y} p = q$

For simplicity we will write $\mathrm{gen}(a) : F(A)$ as $a : F(A)$.

A presented group is a quotient of the free group with many relations between generators. The general schemata of higher inductive types will be(suppose there are finite many relations):

- All constructors of $F(A)$
- For each relation, a path $a_1^{\pm 1} ... a_n^{\pm 1} = e$, where a_i are generators.

Let us consider some examples to close this section.

Definition 4.1.5 The **cyclic group** C_n of order n can be defined as a higher inductive type with a parameter $n : \mathbb{N}$:

- a generator $1_n : C_n$
- a unit $0_n : C_n$
- All group operations and axioms and the 0-truncation
- a path $(1_n)^n = 0_n$

where $(\cdot)^n$ denotes the n-times composition of the multiplication, which can be easily defined by induction on $\mathbb{N}$.

There are many ways to define the integer type and a group structure on it. For example, we can define $\mathbb{Z}$ just as a normal inductive type:

Definition 4.1.6 The **integer** is an inductive type generated by the followiing three constructors:

- a point $0 : \mathbb{Z}$
- a map $\mathrm{pos} : \mathbb{N} \to \mathbb{Z}$
- a map $\mathrm{neg} : \mathbb{N} \to \mathbb{Z}$

We could define the addition on $\mathbb{Z}$ by induction. The readers will soon discover that they are stuck in so many case analysis just when they try to prove the group axioms. For the sole purpose in this thesis however, this definition is enough.

Alternatively in group theory we can also regard $\mathbb{Z}$ as the free group $F(1)$. This definition is straightforward, but lacks the good property of doing inductions. Moreover, it will be hard to decide the canonical form of terms computationally.

A perhaps better solution is to consider $\mathbb{Z}$ as a quotient of $\mathbb{N} \times \mathbb{N}$ with the following relation:

$$(a, b) \sim (c, d) \Leftrightarrow (a + d = b + c) \tag{4.1.3}$$

The monoid operation is defined as $(a, b) + (c, d) := (a + c, b + d)$ and the unit element is $(0, 0)$. The inverse of (a, b) is thus (b, a). All these operations are well defined because the projection $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ is a surjection.

This is also called the **Grothendieck group** of $\mathbb{N}$. Using the universal property of the quotients, we can find a good induction principle for this type that mimics the one of our naive definition.

We mention that it is now possible to define a power function $G \times \mathbb{Z} \to G$ by induction on $\mathbb{Z}$. This observation is helpful by calculating the fundamental group of $\mathbb{S}^1$ in Theorem 4.3.4.

4.2 Loop spaces and homotopy groups

We can focus on the subcategory of pointed types now, this imitates the concept of pointed spaces. We see that in pointed types many objects of homotopy theory are now synthetic.

Recall that in Definition 2.1.8 we have defined the loop space of a pointed type. It is an analogy of the homotopy groups' definition in classic algebraic topology. However, in order to introduce a notion of homotopy groups that matches our usual set-theoretic description of such objects, we need to truncate it into 0-types.

Definition 4.2.1 The n-th **homotopy group** $\pi_n(A)$ of a pointed type A for $n \geq 0$ is the 0-truncation of the n-th loop space $|\Omega^n(A, a)|_0$.

Proposition 4.2.2 $\pi_n(A)$ is a group for $n \geq 1$.

Proof. By universal property of truncations we can define the group operation $u \cdot v : \pi_n(A) \times \pi_n(A) \to \pi_n(A)$ by assuming $u = |p|_0$ and $v = |q|_0$. For $n \geq 1$ p and q are paths. Then we define $|p|_0 \cdot |q|_0 := |p \cdot q|_0$, where $(\cdot)$ means the path concatenation. We also define $|p|_0^{-1} := |p^{-1}|_0$ By Proposition 2.1.2 we know immediately these operators fulfill the group axioms. ∎

Proposition 4.2.3 For $A : \mathcal{U}_\bullet$ we have $\pi_n(A) = \Omega^n(|A|_n, |a|_n)$ for $n \geq 1$.

Proof. We prove it by induction. For $n = 1$ we show in general $|\Omega(A, a)|_k = \Omega(|A|_{k+1}, |a|_{k+1})$. By definition we need to show $|a = a|_k \simeq (|a|_{k+1} = |a|_{k+1})$.

We will use the so called encode-decode method to complete the proof. In general, we define a whole type family code to describe the behavior of the type, on which it is very hard or impossible to do induction. Then we can find two sections encode and decode of the type family and show they are quasi inverses.

We define $\mathrm{code} : |A|_{k+1} \to |A|_{k+1} \to \mathcal{U}$ with $\mathrm{code}(|x|_{k+1}, |y|_{k+1}) :\equiv |x = y|_k$. It is well-defined because $|x = y|_k$ is also $k+1$-truncated, and we can use the universal property of truncations to reduce the terms of $|A|_{k+1}$.

Now we define the $\mathrm{decode} : \mathrm{code}(u, v) \to (u = v), \forall u, v : |A|_{k+1}$ as $\mathrm{decode}(|p|_k) :\equiv (|-|_{k+1})_*(p)$. Conversely we have a map $\mathrm{encode} : (u = v) \to \mathrm{code}(u, v)$ by $\mathrm{encode}(p) :\equiv \mathrm{transport}^{u \mapsto \mathrm{code}(u,v)}(p, r(u))$, where $r : \prod_{u:\, |A|_{k+1}} \mathrm{code}(u, u)$ is given by $r(|x|_{k+1}) :\equiv |\mathrm{refl}_x|_k$ from induction.

By construction and computation of transport we see that they are quasi inverses. When we set $u = v = |a|_{k+1}$ we get the desired result.

Finally suppose we have already $|\Omega^n(A, a)|_k = \Omega^n(|A|_{n+k}, |a|_{n+k})$, then

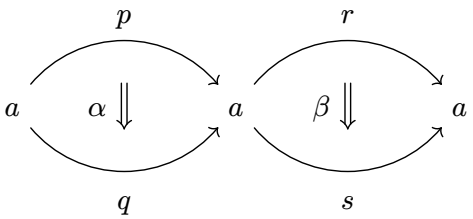
$$\begin{aligned} |\Omega(\Omega^n(A, a))|_k &= \Omega(|\Omega^n(A, a)|_{k+1}) \\ = \Omega(\Omega^n(|A|_{n+k+1}, |a|_{n+k+1})) &= \Omega^{n+1}(|A|_{n+k+1}, |a|_{n+k+1}) \end{aligned} \tag{4.2.1}$$

by induction hypothesis. The proposition follows by setting $k = 0$. ∎

We prove an interesting theorem in traditional algebraic topology.

Theorem 4.2.4 (Eckmann-Hilton) For a pointed type (A, a) the composition of higher paths on $\Omega^n(A, a), n \geq 2$ is commutative. We conclude that the higher homotopy groups are always abelian.

Proof. Consider the following diagram of paths:



where $p, q, r, s : a = a.$ are 1-paths and $\alpha : p = q, \beta : r = s$ are 2-paths. We use the so called "wiskering" as a technique to define horizontal composition of $\alpha \star \beta : p \cdot r = q \cdot s$.

First, we have $\alpha \cdot_r r : p \cdot r = q \cdot r$ by induction on r, we set $\alpha \cdot_r \mathrm{refl}_a :\equiv \mathrm{lu}_p^{-1} \cdot \alpha \cdot \mathrm{ru}_q$ where lu_p and ru_q are defined in Proposition 2.1.2. We can define the left wiskering of β as $q \cdot_l \beta : q \cdot r = q \cdot s$. Induction on q, we set $\mathrm{refl}_a \cdot_l \beta :\equiv \mathrm{lu}_r^{-1} \cdot \beta \cdot \mathrm{ru}_s$. Finally, we define $\alpha \star \beta :\equiv (\alpha \cdot_r r) \cdot (q \cdot_l \beta)$, which has the type $p \cdot r = q \cdot s$.

Analogously we have another composition $\alpha \star' \beta :\equiv (p \cdot_l \beta) \cdot (\alpha \cdot_r s)$. We want to show the two compositions agree with each other with a 3-path, namely $\alpha \star \beta = \alpha \star' \beta$. This can be done by

induction on α and β to reduce them into refl_p and refl_r. A further induction on p and r reduce everything to refl_a and in this case we define $\mathrm{refl}_{\mathrm{refl}_a} \star \mathrm{refl}_{\mathrm{refl}_a} = \mathrm{refl}_{\mathrm{refl}_a} \star' \mathrm{refl}_{\mathrm{refl}_a} := \mathrm{refl}_{\mathrm{refl}_{\mathrm{refl}_a}}$.
Now back to our $\Omega^2(A, a)$ where $p, q, r, s := \mathrm{refl}_a$. In this case

$$\begin{aligned} & \alpha \star \beta \\ = & (\alpha \cdot_r \mathrm{refl}_a) \cdot (\mathrm{refl}_a \cdot_l \beta) \\ = & \mathrm{lu}^{-1}_{\mathrm{refl}_a} \cdot \alpha \cdot \mathrm{ru}_{\mathrm{refl}_a} \cdot \mathrm{lu}^{-1}_{\mathrm{refl}_a} \cdot \beta \cdot \mathrm{ru}_{\mathrm{refl}_a} \\ = & \alpha \cdot \beta \end{aligned} \tag{4.2.2}$$

since $\mathrm{lu}_{\mathrm{refl}_a} = \mathrm{refl}_{\mathrm{refl}_a} = \mathrm{ru}_{\mathrm{refl}_a}$ by path induction. Similarly $\alpha \star' \beta = \beta \cdot \alpha$. In conclusion we have $\alpha \cdot \beta = \beta \cdot \alpha$ the desired commutative property.
For $n \geq 3$, we can simply go one step inductively "higher" in the diagram, i.e. replace a with refl_a etc. and the proof follows. ■
We can prove many classical properties of homotopy groups.

Proposition 4.2.5 For $A, B : \mathcal{U}_\bullet$ we have $\pi_n(A \times B) = \pi_n(A) \times \pi_n(B)$.

Proof. The proof consists of two parts. We will show that $\Omega(A \times B, (a, b)) = \Omega(A, a) \times \Omega(B, b)$ at first. This is true using the path induction on products. By induction on n we get $\Omega^n(A \times B, (a, b)) = \Omega^n(A, a) \times \Omega^n(B, b)$.
It remains to show that the n-truncation preserves products, then we have $\pi_n(A \times B) = |\Omega^n(A, a) \times \Omega^n(B, b)|_0 = |\Omega^n(A, a)|_0 \times |\Omega^n(B, b)|_0 = \pi_n(A) \times \pi_n(B)$.
Let C be an n-type, then $B \to C$ is also an n-type. We thus have

$$\begin{aligned} (|A|_n \times |B|_n \to C) & \simeq (|A|_n \to (|B|_n \to C)) \\ & \simeq (|A|_n \to (B \to C)) \\ & \simeq (A \to (B \to C)) \\ & \simeq (A \times B \to C) \end{aligned} \tag{4.2.3}$$

by Proposition 3.4.4. By universal property of truncations we thus have $|A|_n \times |B|_n \simeq |A \times B|_n$ as desired. ■

Proposition 4.2.6 If $A : \mathcal{U}_\bullet$ is an n-type, then $\forall k > n$ we have $\pi_k(A) = 1$.

Proof. By definition of n-types we know immediately $\Omega^k(A, a)$ is an $(n - k)$-type if A is an n-type. But $(n - k) \leq -1$ and it's inhabited by the reflexivity, hence it must be contractible, we have then $\Omega^k(A, a) = 1$ and $\pi_k(A) = |1|_0 = 1$. ■

Proposition 4.2.7 If $A : \mathcal{U}_\bullet$ is n-connected then $\forall k \leq n$ we have $\pi_k(A) = 1$.

Proof. Using Proposition 4.2.3 we know $\pi_k(A) = \Omega^k(|A|_k)$. Since A is n-connected we have $|A|_n = 1$ and since truncation is cumulative we have $|A|_k = \|A|_n|_k$ when $k \leq n$. Thus $\pi_k(A) = \Omega^k(1) = 1$ as 1 is contractible. ■

4.3 Homotopy group of CW-complexes

We now calculate homotopy groups of certain CW-complexes. An easy to handle example is the sphere.

Proposition 4.3.1 If $X : \mathcal{U}$ is n-connected, then ΣX is $n+1$-connected.

Proof. We need to show $\Sigma X \to 1$ is an $(n+1)$ connected map, i.e. $|\Sigma X|_{n+1}$ is contractible. By the definition we know that $\Sigma X \simeq 1 \sqcup_X 1$. Since 1 is the only contractible type up to equivalence, by Proposition 3.2.7 if 1 is also the pushout of $\mathcal{D} : 1 \leftarrow |X|_{n+1} \to 1$ then we have $1 = |1 \sqcup_X 1|$. First we need to verify 1 is a cocone. Now consider $\mathrm{id} : 1 \to 1$, it's trivial that $\forall x : |X|_{n+1}, \mathrm{id}(*) = \mathrm{id}(*)$ by refl_*. Let $E : \mathrm{cocone}(\mathcal{D})$ be an $(n+1)$-type with $p, q : 1 \to E$. Since $(1 \to E) \simeq E$ we have now $x, y : E$ and $f : |X|_{n+1} \to (x = y)$ by expanding the definition of cocone. It remains to show that $(x = y)_E \simeq (|X|_{n+1} \to (x = y))$ then we can define $h : 1 \to E$ just to be $h(*) := x$, this commutes thus with id and p, q.
But since X is n-connected, then so is $|X|_{n+1}$ as the truncation is cumulative. Now by Proposition 3.4.7 $\left((|X|)_{n+1} \to (x = y)\right) \simeq (x = y)$ because E is an $n+1$-type and $(x = y)$ is an n-type for all $x, y : E$. ■

Corollary 4.3.2 The n-th sphere $\mathbb{S}^n$ is $(n-1)$-connected.

Proof. We prove by induction on n. If $n = 2$, we know that $\mathbb{S}^0$ is inhabited, thus $|\mathbb{S}^0|_{-1}$ is contractible. Now suppose $\mathbb{S}^n$ is $(n-1)$-connected, we have $\mathbb{S}^n = \Sigma\mathbb{S}^{n-1}$ is n-connected by the previous proposition. ■

Corollary 4.3.3 $\pi_k(\mathbb{S}^n) = 1$ for $k < n$.

Proof. By combining Corollary 4.3.2 and Proposition 4.2.7. ■
A non-trivial example is the fundamental group of the circle.

Theorem 4.3.4 $\pi_1(\mathbb{S}^1) = \mathbb{Z}$.

Proof. This is an instance of the encode-decode method. We will show a stronger result that $\Omega(\mathbb{S}^1) = \mathbb{Z}$, this also implies the higher homotopy group of $\mathbb{S}^1$ is trivial.
We will use the following definition of $\mathbb{S}^1$, a simple geometric proof shows it is equivalent to $\Sigma\mathbb{S}^0$.

- a point $\mathrm{base} : \mathbb{S}^1$
- a path $\mathrm{loop} : \mathrm{base} = \mathrm{base}$

Now we define the universal cover of $\mathbb{S}^1$ to be a type family $\mathrm{code} : \mathbb{S}^1 \to \mathcal{U}$ such that $\mathrm{code}(\mathrm{base}) = \mathbb{Z}$ and $\mathrm{code}_*(\mathrm{loop}) = \mathrm{ua}(\mathrm{succ})$.

In order to prove $(\mathrm{base} = \mathrm{base}) \simeq \mathbb{Z}$, it is better to consider all fibrations of the cover. That means, we need to define a family of equivalences $\prod_{x:\mathbb{S}^1}(\mathrm{base} = x) \to \mathrm{code}(x)$ and $\prod_{x:\mathbb{S}^1} \mathrm{code}(x) \to (\mathrm{base} = x)$ and apply it to $\mathrm{base} : \mathbb{S}^1$.

With the computation rule of the transport functor, we can show that $\mathrm{transport}^{\mathrm{code}}(\mathrm{loop}, x) = \mathrm{succ}(x)$ and $\mathrm{transport}^{\mathrm{code}}(\mathrm{loop}^{-1}, x) = \mathrm{pred}(x)$ since succ and pred are quasi inverses.

We define $\mathrm{encode} : \prod_{x:\mathbb{S}^1}(\mathrm{base} = x) \to \mathrm{code}(x)$ with $\mathrm{encode}(x, p) :\equiv \mathrm{transport}^{\mathrm{code}}(p, 0)$. On the other hand, we define $\mathrm{decode} : \prod_{x:\mathbb{S}^1} \mathrm{code}(x) \to (\mathrm{base} = x)$ by induction on $\mathbb{S}^1$, we define the function $\mathrm{loop}^- : \mathrm{code}(\mathrm{base}) \to (\mathrm{base} = x)$ as the power function with respect to the path concatenation.

We also need to find a path of loop^- to itself that lies over loop, i.e. a path

$$\mathrm{transport}^{\mathrm{decode}}(\mathrm{loop}, \mathrm{loop}^-) = \mathrm{loop}^- \tag{4.3.1}$$

This is by computation on transport and the fact $\mathrm{transport}^{\mathrm{code}}(\mathrm{loop}^{-1}) = \mathrm{pred}$.

We show $\forall x : \mathbb{S}^1$ and $p : \mathrm{base} = x$ we have $\mathrm{decode}_x(\mathrm{encode}_x(p)) = p$. By path induction we may assume $p :\equiv \mathrm{refl}_{\mathrm{base}}$, in this case we have $\mathrm{encode}_{\mathrm{base}}(\mathrm{refl}_{\mathrm{base}}) = \mathrm{transport}(\mathrm{refl}_{\mathrm{base}}, 0) = 0$ and $\mathrm{decode}_{\mathrm{base}}(0) = \mathrm{refl}_{\mathrm{base}}$ by definition.

Now we show $\forall x : \mathbb{S}^1$ and $c : \mathrm{code}(x)$ we have $\mathrm{encode}_x(\mathrm{decode}_x(c)) = c$. By induction on $\mathbb{S}^1$, we give the case for $x :\equiv \mathrm{base}$ and the functoriality follows from the fact that $\mathbb{Z}$ is a set and the induced path of loop is a 2-path on $\mathbb{Z}$, which is of course the reflexivity.

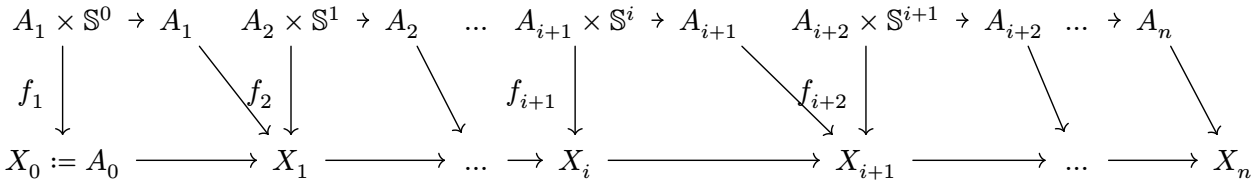
In this case we need to show $\mathrm{encode}_{\mathrm{base}}(\mathrm{decode}_{\mathrm{base}}(n)) = n, \forall n \in \mathbb{Z}$. We do an induction on $\mathbb{Z}$ and after case analysis and computation, we know this is true.

As from our construction, $\mathrm{encode}_{\mathrm{base}}$ and $\mathrm{decode}_{\mathrm{base}}$ preserves the group operations by sending addition to concatenation and vice versa, hence $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ as a group. ■

Now we will turn to the general theory of cell complexes. A cell complex consists of n-dimensional discs, which are als called cells, with their boundaries attached on the $(n-1)$-cells. The cell complex is finite in both senses that the dimension is finite and the number of n-cells for each n is also finite.

We can give a general schemata for finite cell complexes via higher inductive types.

Definition 4.3.5 Let A_i be index types, a finite n-dimensional **cell complex** is an iterated pushout in such diagram:



where X_n denotes the n-skeleton of the complex, i.e. all cells under dimension n and f_i is the product of attaching maps from the boundaries of n-cells onto $n-1$-skeleton.

Remark 4.3.6 A finite cell complex in HoTT can be presented as a higher inductive types, where all n-cells are n-paths, and the type of n-path is the partition of the attaching map into source and target. Concretely, a higher inductive type X with:

- some points $X_0 : A_0 \to X$
- some paths $X_1 : A_1 \to (X_0(a) = X_0(b))$, a, b are related to the attaching map.

…

- some i-paths $X_i : A_i \to (f_i(X_{i-1}) = f_i'(X_{i-1}))$, f produces a composition of $i-1$-paths.

…

We will see the n-th homotopy group of a cell complex only depends on its $n+1$-skeleton.

Proposition 4.3.7 Let X be a finite cell complex, then $\pi_n(X) = \pi_n(X_{n+1})$.

Proof. By Proposition 4.2.3 we need to show $\Omega^n(|X|_n) = \Omega^n(|X_{n+1}|_n)$. It suffices to show $|X|_n \simeq |X_{n+1}|_n$. We consider the following diagram.

$$\begin{array}{ccccccc}
A_{n+1} \times |\mathbb{S}^n|_n & \to & A_{n+1} & & A_{n+2} \times |\mathbb{S}^{n+1}|_n & \to & A_{n+2} \\
\downarrow & & \searrow & & \downarrow & & \searrow \\
\ldots \longrightarrow |X_n|_n & & \longrightarrow & & |X_{n+1}|_n & \xrightarrow{i} & |X_{n+2}|_n
\end{array}$$

We know $\mathbb{S}^{n+1}$ is n-connected by Corollary 4.3.2, thus $|\mathbb{S}^{n+1}|_n = 1$. By definition, the pushout of the diagram $|X_{n+1}|_n \leftarrow 1 \to 1$ is the wedge sum $|X_{n+1}|_n \vee 1$, which is equivalent to $|X_{n+1}|_n$ since 1 is contractible. But all A_is are finite, thus by universality we have $i : |X_{n+1}|_n \to |X_{n+2}|_n$ an equivalence. Finally by inductively combining all equivalences we get $|X_{n+1}|_n \simeq |X|_n$. ∎

We can also understand this property from the explicit definition of cell complexes as a higher inductive type. After n-truncation all higher paths become trivial, which gives an intuitional way to see the equivalence between the whole complex and the $(n+1)$-skeleton.

In classical algebraic topology this proposition is proven using cellular approximation theorem. But here things become natural from the definition as we have defined the n-truncation somewhat like the postnikov section of a higher inductive type. At the same time, a big part of the higher inductive types could be considered as a cell complex. This analogy is essential in this synthetic homotopy theory.

An example of cell complexes is the torus T^2 defined by:

- a point $\mathrm{base} : T^2$
- two paths $p, q : \mathrm{base} = \mathrm{base}$
- a homotopy $r : p \cdot q = q \cdot p$

This definition can be easily generalized to the compact surfaces with genus, which characterized all compact orientable 2-manifolds up to homotopy.

Definition 4.3.8 A **compact 0-connected and orientable surface** $S(g)$ with genus g is generated as the following higher inductive type:

- a point $\mathrm{base} : S(g)$
- some paths $p_i : \mathrm{base} = \mathrm{base}, \forall 1 \leq i \leq 2g$
- a homotopy $\mathrm{rel} : p_1 \cdot p_2 ... \cdot p_{2g} = p_{2g} \cdot p_{2g-1} \cdot ... p_1$

This type definition may be simplified using a indexed type as the argument instead of giving all paths, i.e. 1-cells explicitly. Note that the indexed type should not have a group structure, a nice candidate will then be the finite set Fin in dependent type theory.

Theorem 4.3.9 $\pi_1(S(g)) = G :=< a_1, ..., a_{2g} \mid a_1 ... a_{2g} a_1^{-1} ... a_{2g}^{-1} >$.

Proof. We prove the equivalence without the truncation. By definition we know that $\Omega(S(g), \mathrm{base}) = (\mathrm{base} = \mathrm{base}, \mathrm{refl}_{\mathrm{base}})$. An induction on the loop space shows that it suffices to define a function $f : \Omega(S(g)) \to G$ which sends p_i to a_i and the homotopy rel to the relation $a_1 ... a_{2g} a_1^{-1} ... a_{2g}^{-1} = e$, which is equal to the equality $a_1 ... a_{2g} = a_{2g} ... a_1$ by group axioms.
By induction on G we get a function $g : G \to \Omega(S(g))$ by sending a_i to p_i and the relation to the constructor rel.
Clearly f and g are quasi inverses and they preserve the group operations canonically. ∎

Remark 4.3.10 To formalize this proof it needs actually some more constructions. We must define the represented group precisely as a higher inductive type and use the encode-decode method to define our map $f : \Omega(S(g)) \to \mathrm{list}(A)$ properly.

Of course when we consider the torus as a special case with genus 1, there is another better characterization.

Corollary 4.3.11 $\pi_1(T^2) = \mathbb{Z} \times \mathbb{Z}$ where $T^2 := S^1 \times S^1$ the torus.

Proof. By Proposition 4.2.5 and Theorem 4.3.4. ∎

A nice example of the non-orientable surface will definitely be $\mathbb{RP}^2$, which can be defined as the following higher inductive type:

- a point $\mathrm{base} : \mathbb{RP}^2$
- a path $q : \mathrm{base} = \mathrm{base}$
- a homotopy $r : q \cdot q = \mathrm{refl}_{\mathrm{base}}$

Proposition 4.3.12 $\pi_1(\mathbb{RP}^2) = C_2$

Proof. Encode-decode method. This is completely analogous to the Theorem 4.3.4. In addition that we will map $\mathrm{code}_*(r)$ to the relation $1_2 \cdot 1_2 = 0_2$ and do some calculations on 2-paths. ■

4.4 Stable homotopy and cohomology theory

A classical Result from Brown states that any reduced cohomology theory can be represented by the homotopy classes to a concrete space(see [Hat02] 4E.1 for a proof). This motivates us to define a cohomology theory in a synthetic way. Just like other results in HoTT, this theory is independent of realizations.

We will use higher inductive types to define Eilenberg-MacLane spaces at first.

Definition 4.4.1 The **Eilenberg-MacLane space** $K(G,1)$ for any group G is defined by:

- a point $\mathrm{base} : K(G,1)$
- a map $\mathrm{bonq} : G \to (\mathrm{base} = \mathrm{base})$
- a path $\mathrm{unit} : \mathrm{bonq}(e) = \mathrm{refl}_{\mathrm{base}}$
- a dependent map $\mathrm{homo} : \prod_{x,y:G} (\mathrm{bonq}(x \cdot y) = \mathrm{bonq}(x) \cdot \mathrm{bonq}(y))$
- a dependent map $\mathrm{inv} : \prod_{x:A} \left(\mathrm{bonq}(x^{-1}) = \mathrm{bonq}(x)^{-1}\right)$
- a dependent map $\mathrm{trunc} : \prod_{x,y:K(G,1)} \prod_{p,q:x=y} \prod_{r,s:p=q} (r = s)$

The last constructor ensures that $K(G,1)$ is indeed an 1-type. The only thing we need to show is the following lemma.

Lemma 4.4.2 $\pi_1(K(G,1)) = G$

Proof. From [LF14]. We use the encode-decode method again. Define the type family $\mathrm{code} : K(G,1) \to \mathcal{U}$ such that $\mathrm{code}(\mathrm{base}) :\equiv G$ and $\mathrm{transport}^{\mathrm{code}}(\mathrm{bonq}(x), y) = x \cdot y$. The two sections are given by

$$\begin{aligned} \mathrm{encode} &: \prod_{a:K(G,1)} (\mathrm{base} = a) \to \mathrm{code}(a) \\ \mathrm{encode}_a(p) &:\equiv \mathrm{transport}^{\mathrm{code}}(p, e) \\ \mathrm{decode} &: \prod_{a:K(G,1)} \mathrm{code}(a) \to (\mathrm{base} = a) \\ \mathrm{decode}_{\mathrm{base}}(x) &:\equiv \mathrm{bonq}(x) \end{aligned} \tag{4.4.1}$$

The rest is just plain computation. The reader will immediately notice the analogous part of the proof with Theorem 4.3.4. This is justified by Example 4.4.4. ■

The general Eilenberg-MacLane space of degree n is defined then inductively.

Definition 4.4.3 For an abelian group G and $n \geq 2$, the **Eilenberg-MacLane space** $K(G,n)$ is defined by:

$$K(G, n+1) :\equiv |\Sigma(K(G,n))|_{n+1} \tag{4.4.2}$$

Example 4.4.4 From the previous section we know that $K(\mathbb{Z}, 1) \simeq \mathbb{S}^1$.

Proposition 4.4.5 For $n \geq 1$ the $K(G, n)$ we have defined is indeed an Eilenberg-MacLane space, i.e. $\pi_n(K(G,n)) = G$ and $\pi_k(K(G,n)) = 1$ for $k \neq n$.

This proof of proposition will use the Freudental suspension theorem, which is proven using encode-decode method.

Theorem 4.4.6 (Freudental suspension theorem) Let $X : \mathcal{U}_\bullet$ be n-connected, then we have the equivalence $\forall k \leq 2n$:

$$|X|_k \simeq |\Omega(\Sigma X)|_k \tag{4.4.3}$$

Proof of Proposition 4.4.5. For $n = 1$, consider Lemma 4.4.2, it remains to show $\pi_k(K(G,1)) = 1, k > 1$, but this follows directly from Proposition 4.2.7 since $K(G,1)$ is an 1-type.
Observe that $K(G,1)$ is 0-connected since it has just a point base, thus $K(G,n)$ is $n-1$-connected by Proposition 4.3.1, and by Proposition 4.2.7 $\pi_k(K(G,n)) = 1$ for $k \leq n-1$. From then definition we know that $K(G,n)$ is an n-type, thus $\pi_k(K(G,n)) = 1$ for $k > n$. We show $\pi_n(K(G,n)) = G$ by induction.
We have

$$\begin{aligned} \Omega^{n+1}(K(G,n+1)) &= \Omega^{n+1}(|\Sigma(K(G,n))|_{n+1}) \\ \Omega^n(|\Omega\Sigma(K(G,n))|_n) &= \Omega^n(|K(G,n)|_n) \end{aligned} \tag{4.4.4}$$

by Proposition 4.2.3 and the Freudental suspension theorem respectively. ∎

Definition 4.4.7 Let $n \geq 1$, the **reduced cohomology** $H^n(A, G)$ for a type $A : \mathcal{U}_\bullet$ with coefficients in G is defined by:

$$H^n(A, G) :\equiv |(A* \to K(G,n))|_0 \tag{4.4.5}$$

Proposition 4.4.8 $H^n(\mathbb{S}^n, G) = G$ and $H^k(\mathbb{S}^k, G) = 1$ for $k \neq n$.

Proof. By definition we get $H^k(\mathbb{S}^n, G) = |\mathbb{S}^n * \to K(G,k)|_0 = |\Omega^n(K(G,k))|_0$. The statement follows from the definition of Eilenberg-MacLane space. ∎
To compute more complicated cohomology, one must generalize the above construction and dive into the calculation of spectral sequences. The Freudental suspension theorem is a good start point, but we still await a fully operational stable category in HoTT.
The discussion is beyond the scope of this thesis and we will end the chapter here.

Bibliography

[Mar84] P. Martin-Löf, *Intuitionistic Type Theory*, Vol. 1 (Bibliopolis, Naples, 1984)

[AW09] S. Awodey and M. A. Warren, *Homotopy Theoretic Models of Identity Types*, in Vol. 146 (2009), pp. 45–55

[Voe10] V. Voevodsky, *Univalent Foundations Project*, (2010)

[LS20] P. L. Lumsdaine and M. Shulman, *Semantics of Higher Inductive Types*, in Vol. 169 (2020), pp. 159–208

[KL21] K. Kapulkin and P. L. Lumsdaine, *The Simplicial Model of Univalent Foundations (After Voevodsky)*, Journal of the European Mathematical Society **23**, 2071 (2021)

[Uni13] Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics* (Institute for Advanced Study, Princeton, New Jersey, 2013)

[Gro21] A. Grothendieck, *Pursuing Stacks*, (2021)

[BCH19] M. Bezem, T. Coquand, and S. Huber, *The Univalence Axiom in Cubical Sets*, Journal of Automated Reasoning **63**, 159 (2019)

[VMA19] A. Vezzosi, A. Mörtberg, and A. Abel, *Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types*, **3**, (2019)

[Doo15] F. van Doorn, *Constructing the Propositional Truncation Using Non-Recursive Hits*, (2015)

[BR17] U. Buchholtz and E. Rijke, *The Real Projective Spaces in Homotopy Type Theory*, in *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science* (IEEE Press, Reykjavík, Iceland, 2017)

[Hat02] A. Hatcher, *Algebraic Topology* (Cambridge University Press, Cambridge, UK, 2002)

[LF14] D. R. Licata and E. Finster, *Eilenberg-Maclane Spaces in Homotopy Type Theory*, in *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (Association for Computing Machinery, Vienna, Austria, 2014)